C 8

```
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRRRRRRRRRRR           III        VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD         DDD  RRRRRRRRRRRR           III        VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD         DDD  RRRRRRRRRRRR           III        VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD         DDD  RRR   RRR              III        VVV         VVV  EEE              RRR   RRR
DDD         DDD  RRR   RRR              III        VVV         VVV  EEE              RRR   RRR
DDD         DDD  RRR   RRR              III        VVV         VVV  EEE              RRR   RRR
DDD         DDD  RRR      RRR           III           VVV   VVV     EEE              RRR      RRR
DDD         DDD  RRR      RRR           III           VVV   VVV     EEE              RRR      RRR
DDD         DDD  RRR      RRR           III           VVV   VVV     EEE              RRR      RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII           VVV        EEEEEEEEEEEEEEE  RRR         RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII           VVV        EEEEEEEEEEEEEEE  RRR         RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII           VVV        EEEEEEEEEEEEEEE  RRR         RRR
```

```
**FILE**ID**TUDRIVER                                 G  7





TTTTTTTTTT  UU      UU  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
TTTTTTTTTT  UU      UU  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
    TT      UU      UU  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
    TT      UU      UU  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
    TT      UU      UU  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
    TT      UU      UU  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
    TT      UU      UU  DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
    TT      UU      UU  DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
    TT      UU      UU  DD      DD  RR  RR          II      VV      VV  EE          RR  RR
    TT      UU      UU  DD      DD  RR  RR          II      VV      VV  EE          RR  RR
    TT      UU      UU  DD      DD  RR    RR        II        VV  VV    EE          RR    RR      ....
    TT      UU      UU  DD      DD  RR    RR        II        VV  VV    EE          RR    RR      ....
    TT      UUUUUUUUUU  DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR    ....
    TT      UUUUUUUUUU  DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR    ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
0000            .TITLE  TUDRIVER - TAPE CLASS DRIVER
0000            .IDENT  'V04-000'
0000
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000    ; Robert Rappaport  16-June-1982
0000    ;
0000    ;  TAPE CLASS DRIVER
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;       V03-161 ROW0398         Ralph O. Weber          21-JUL-1984
0000    ;               Setup use of class driver write-lock bit in UCB$W_DEVSTS.
0000    ;
0000    ;       V03-160 ROW0396         Ralph O. Weber          21-JUL-1984
0000    ;               Setup automatic detection of density after an operation which
0000    ;               moves the tape position off of the BOT.
0000    ;
0000    ;       V03-159 ROW0395         Ralph O. Weber          21-JUL-1984
0000    ;               Make changes which setup "normal" MSCP command timeout
0000    ;               algorithm before calls to DUTU$POLL_FOR_UNITS and
0000    ;               BRING_UNIT_ONLINE.  Also setup use of DAP CDRP by both
0000    ;               DUTU$POLL_FOR_UNITS and BRING_UNIT_ONLINE.
0000    ;
0000    ;       V03-158 ROW0394         Ralph O. Weber          20-JUL-1984
0000    ;               Remove DPT_STORE setting of ACL queue present bit in the ORB.
0000    ;               This should improve performance on devices which do not really
0000    ;               have an ACL queue in their device protection ORB.
0000    ;
0000    ;       V03-157 ROW0393         Ralph O. Weber          20-JUL-1984
0000    ;               Add media-id to device type translation table entries for the
0000    ;               TA78, TK50, and TA81.
0000    ;
0000    ;       V03-156 ROW0387         Ralph O. Weber          8-JUL-1984
0000    ;               Setup use of DUTU$RECONN_LOOKUP and DUTU$DRAIN_CDDB_CDRPQ.
```

```
0000   ;
0000   ;       V03-155 ROW0369         Ralph O. Weber          6-JUL-1984
0000   ;               Change DU$RE_SYNCH to not do MRESET/MSTART to MSCP servers and
0000   ;               then wait for something to happen.  Quite possibly, nothing
0000   ;               ever will happen in such cases.  Proceeding directly to the
0000   ;               DISCONNECT is the correct action.  This is being done now so
0000   ;               that it will not be forgotten when as and if we make a tape
0000   ;               MSCP server.
0000   ;
0000   ;       V03-154 ROW0382         Ralph O. Weber          22-JUN-1984
0000   ;               Change START_PACKACK so the an exclusive access online command
0000   ;               is sent only the multihost controllers.  For other controllers,
0000   ;               just sent an online.
0000   ;
0000   ;       V03-153 ROW0361         Ralph O. Weber          5-MAY-1984
0000   ;               Setup use of new class driver common DAP processing in
0000   ;               DUTU$DODAP.  The new routine is designed to eliminate multiple
0000   ;               concurrent DAP threads which are known to crash systems.
0000   ;
0000   ;       V03-152 ROW0354         Ralph O. Weber          30-APR-1984
0000   ;               Add setting for DEV$M_NNM in DEVCHAR2 to indicate that tape
0000   ;               class driver devices use NODENAME$DDCN device names.
0000   ;
0000   ;       V03-151 ROW0353         Ralph O. Weber          30-APR-1984
0000   ;               Correct message type constant input to ERL$LOGMESSAGE from
0000   ;               EMB$C_DM (for disks) to EMB$C_TM (for tapes).
0000   ;
0000   ;       V03-150 ROW0350         Ralph O. Weber          23-APR-1984
0000   ;               Correct more problems causing multiple trips through
0000   ;               END_SINGLE_STREAM, with the attendent bugchecks.  First, clear
0000   ;               CDDB$V_SNGLSTRM upon entry to DU$CONNECT_ERR.  Second, protect
0000   ;               the SCS$UNSTALLUCB loop in END_SINGLE_STREAM from possible
0000   ;               connection failures during execution of the loop.
0000   ;
0000   ;       V03-149 LMP0237         L. Mark Pilant,         19-Apr-1984  11:25
0000   ;               Initialize the template ORB.
0000   ;
0000   ;       V03-148 ROW0347         Ralph O. Weber          11-APR-1984
0000   ;               Cause MT$V_HWL to be cleared when tape is not write locked and
0000   ;               whenever an AVAILABLE command is sent to the server.
0000   ;
0000   ;       V03-147 ROW0339         Ralph O. Weber          9-APR-1984
0000   ;               Setup use of common invalid command processing routines
0000   ;               (macros).  This replaces the old ''form the original MSCP
0000   ;               command packet by hand'' algorithm with a ''repeat the code
0000   ;               which formed the original MSCP command'' algorithm.  The cost
0000   ;               is a single, hardly ever taken BLBS in the mainline read/write
0000   ;               code path.  The savings are elimination of having to duplicate
0000   ;               command packet setup changes in the invalid command case,
0000   ;               hundreds of bytes of code, and a not inconsequential amount of
0000   ;               static storage.
0000   ;
0000   ;       V03-146 ROW0338         Ralph O. Weber          7-APR-1984
0000   ;               Setup use of DO_ACTION macro to replace INTERPRET_ACTION_TABLE.
0000   ;               Start using IF_MSCP where only success or failure of an MSCP
0000   ;               command is being tested.  Setup use of ACTION_ENTRY END to end
0000   ;               action tables.  Remove action table interpretation routines;
```

```
0000  ;         they are now in DUTUSUBS.
0000  ;
0000  ;  V03-145 ROW0335           Ralph O. Weber            4-APR-1984
0000  ;         > Correct positioning of DPT_STORE REINIT and add note that
0000  ;           reinit is not significant because driver is not reloadable.
0000  ;         > Add use of DUTU$UNITINIT.  Basicly, this permits future use
0000  ;           of TMSCP devices for booting.
0000  ;         > Remove usage of allocation class value in the SCS connect
0000  ;           accept message.  All MSCP servers now supply that
0000  ;           information in the Set Controller Characteristics command
0000  ;           end packet.
0000  ;         > Eliminate bug check for IO$_READLBLK and IO$_WRITELBLK.
0000  ;           Make these functions produce SS$_ILLIOFUNC status instead.
0000  ;           Also change function dispatcher to use DISPATCH macro.
0000  ;         > Add processing for IO$M_INHRETRY.
0000  ;         > Add the multi-host progress counter handling proposed by the
0000  ;           HSC implementors to TU$TMR.  This algorithm simplifies
0000  ;           handling of the case where the MSCP server is busy on an
0000  ;           older command from another host.
0000  ;
0000  ;  V03-144 ROW0331           Ralph O. Weber            31-MAR-1984
0000  ;         Setup use of common cancel support in DUTUSUBS.  Also make
0000  ;         functions which use multiple MSCP commands check for cancel
0000  ;         after each MSCP command and perform cancel if necessary.
0000  ;
0000  ;  V03-143 ROW0328           Ralph O. Weber            21-MAR-1984
0000  ;         Correct bugs in ROW0319 which caused it to incorrectly miss
0000  ;         the end of the CDDB UCB chain.
0000  ;
0000  ;  V03-142 ROW0324           Ralph O. Weber            12-MAR-1984
0000  ;         > Correct set mode and set characteristics so that
0000  ;           MSCP$W_FORMAT is zero except when the UCB$L_RECORD is zero.
0000  ;           This brings the driver into conformance with TMSCP
0000  ;           version 1.6.
0000  ;         > Provide for proper setup of the following UCB$L_DEVDEPEND
0000  ;           bits in all cases that I can think of:  MT$V_BOT, MT$V_EOF,
0000  ;           MT$V_EOT, MT$V_HWL, MT$V_LOST, MT$V_SUP_NRZI, MT$V_SUP_PE,
0000  ;           and MT$V_SUP_GCR.
0000  ;         > Fix "detect [EOT" modifier setup so that the modifier is
0000  ;           NEVER set for physical I/O requests.
0000  ;         > Change IOSB status returned when a backwards skip file
0000  ;           encounters the BOT to SS$_NORMAL.
0000  ;
0000  ;  V03-141 ROW0320           Ralph O. Weber            29-FEB-1984
0000  ;         Provide for automatic PACKACK on foreign tapes (DEV$V_FOR set)
0000  ;         whenever a request is received and the UCB$V_VALID bit is
0000  ;         clear.  Build the sequential NOP function into macros so that
0000  ;         its use can be easily duplicated where necessary.
0000  ;
0000  ;  V03-140 ROW0319           Ralph O. Weber            28-FEB-1984
0000  ;         Attempt to eliminate failover to non-operational path by
0000  ;         making clearing of CDDB$V_RECONNECT the last thing done in
0000  ;         END_SINGLE_STREAM.  Also add sanity check that CDDB$V_RECONNECT
0000  ;         is set before it is cleared.
0000  ;
0000  ;  V03-139 ROW0310           Ralph O. Weber            23-FEB-1984
0000  ;         Make IO$_REWINDOFF equivalent to IO$_UNLOAD.
```

```
0000    ;
0000    ;       V03-138 ROW0307         Ralph O. Weber          15-FEB-1984
0000    ;               Fix trace support to work in the common modules environment.
0000    ;               Make RECORD_GETUNIT_CHAR preserve RO.
0000    ;
0000    ;       V03-137 ROW0305         Ralph O. Weber          13-FEB-1984
0000    ;               Fix RO (final IOSB status) corruption problems in successful
0000    ;               IO$_PACKACK processing.
0000    ;
0000    ;       V03-136 ROW0301         Ralph O. Weber          10-FEB-1984
0000    ;               Move clearing of CDDB$V_NOCONN from MAKE_CONNECTION to after
0000    ;               the new connection information has been propogated to all UCBs
0000    ;               in the re-connect code.  While this is not absolutely
0000    ;               necessary here and now, it will provide a useful reminder that
0000    ;               CDDB$V_NOCONN set blocks mount verification attempts and thus
0000    ;               the bit cannot be cleared until connection dependent fields in
0000    ;               all UCBs have been altered to reflect the new connection.
0000    ;
0000    ;       V03-135 ROW0299K(ludge) Ralph O. Weber          9-FEB-1984
0000    ;               This kludge detects a HSC tape server in RECORD_STCON and
0000    ;               forces it to act like a multihost server for allocation class
0000    ;               determination, inspite of the fact that the HSC tape server
0000    ;               does not set the multihost controller flag.  This kludge can
0000    ;               be removed when the HSC tape server sets the multihost
0000    ;               controller flag (as it should).
0000    ;
0000    ;       V03-134 ROW0298         Ralph O. Weber          9-FEB-1984
0000    ;               Setup use of CDRP$W_ENDMSGSIZ to hold the size of an incomming
0000    ;               sequenced message.  This replaces use of CDRP$L_IOST2+2 whose
0000    ;               use causes valuable input information to be overwritten.
0000    ;
0000    ;       V03-133 ROW0297         Ralph O. Weber          7-FEB-1984
0000    ;               Correct confusion between wait count bumped due to a broken
0000    ;               connection and wait count bumped due to a sequential NOP by
0000    ;               introducing a UCB$V_TU_SETNOP bit in device dependent status.
0000    ;
0000    ;       V03-132 ROW0294         Ralph O. Weber          5-FEB-1984
0000    ;               Correct RECORD_STCON setup of allocation class information in
0000    ;               the DDBs to use DDB$L_CONLINK so that only those DDBs on this
0000    ;               connection are effected.
0000    ;
0000    ;       V03-131 ROW0293         Ralph O. Weber          5-FEB-1984
0000    ;               Generally bring tape class driver to same revision level as
0000    ;               disk class driver.  The only exception is that there is no
0000    ;               mount verification and thus thing which depend upon it for
0000    ;               updated operation techniques have been left unchanged.
0000    ;               Replace CDRP$V_ERLOGIP in CDRP$W_STS with CDRP$V_ERLIP in
0000    ;               CDRP$L_DUTUFLAGS.  Setup use of CDDB$V_NOCONN status bit.
0000    ;               Setup use of several routines which have been moved to
0000    ;               DUTUSUBS.
0000    ;
0000    ;       V03-130 ROW0272         Ralph O. Weber          1-JAN-1984
0000    ;               Change START_DAP_THREAD to only send Determin Access Paths
0000    ;               commands for those UCBs which are UCB$V_VALID.  MSCP servers
0000    ;               will ignore DAP commands for units which are not MSCP online,
0000    ;               so why should we send them.  Add block which prevents logging
0000    ;               errors for DAP attention messages to ACCESS_PATH_ATTN.  This
```

```
0000    ;               allows the code which logs DAP attention messages to remain
0000    ;               and to be patched back into existance should it be needed.
0000    ;
0000    ;       V03-129 ROW0270         Ralph O. Weber          1-JAN-1984
0000    ;               Eliminate DRIVER_SEND_MSG_BUF by replacing all calls to it
0000    ;               with SEND_MSCP_MSG_DRIVER.  Change MAKE_CONNECTION to use the
0000    ;               larger of HSTIMEOUT_ARRAY[controller_model] and the controller
0000    ;               timeout value as the final host timeout value for the MSCP Set
0000    ;               Controller Characteristics command.  Setup use of VMS SCS
0000    ;               RECYCL_RSPID and FIND_RSPID_RDTE.  Fix START_SENSECHAR and
0000    ;               START_SENSEMODE to clear the MSCP$M_MD_CLSEX (clear serious
0000    ;               exception modifier) bit, as this modifier is illegal on Get
0000    ;               Unit Status commands.  Make all permanent/DAP CDRP to CDDB
0000    ;               conversions use PERMCDRP_TO_CDDB.
0000    ;
0000    ;       V03-128 ROW0269         Ralph O. Weber          1-JAN-1984
0000    ;               Change DU_CONTROLLER_INIT to use DUTU$CREATE_CDDB.
0000    ;
0000    ;       V03-127 ROW0262         Ralph O. Weber          27-DEC-1983
0000    ;               Move all UCB lookup and creation to DUTUSUBS.  Cleanup
0000    ;               ATTN_MSG processing in TU$IDR.  Implement usage of $DUTUDEF,
0000    ;               all device independent UCB fields, and the IOC$GL_TU_CDDB
0000    ;               listhead.  Replace all DPT_STORE macros which init UCB fields
0000    ;               with INIT_UCB macros.  INIT_UCB initializes both the DPT and
0000    ;               the template UCB.  Its use eliminates possible mismatch of the
0000    ;               two UCB sources as well as some setup code in the controller
0000    ;               initialization routine.  Make driver not reloadable.  Change
0000    ;               POLL_FOR_UNITS to DUTU$POLL_FOR_UNITS.
0000    ;
0000    ;       V03-126 ROW0261         Ralph O. Weber          22-NOV-1983
0000    ;               Move DUMP_COMMAND and DUMP_ENDMESSAGE to DUTUSUBS.  Change
0000    ;               TU$END to DUTU$END so that linking with multiple modules does
0000    ;               not involve a hack.  Do some common path cleanup to speed
0000    ;               passage through the common code paths.  Change subroutine
0000    ;               CALL_SEND_MSG_BUF to SEND_MSCP_MSG macro.  Move INIT_TPLATE_UCB
0000    ;               to DUTULIB (macro library).
0000    ;
0000    ;       V03-125 RLRQBUS         Robert L. Rappaport     16-NOV-1983
0000    ;               Change building of transfer commands MSCP packet so that
0000    ;               PQDRIVER can alter the mapping information during a map
0000    ;               request and have the altered information appear in the MSCP
0000    ;               packet.
0000    ;
0000    ;       V03-124 ROW0258         Ralph O. Weber          17-NOV-1983
0000    ;                       The Paul Painter Memorial Enhancement
0000    ;               Named for one of the unfortunate customers who suffered much
0000    ;               to determine the great UCB$L_MT_RECORD secret while trying to
0000    ;               create a user-written magtape driver, this change eliminates
0000    ;               use of the device dependent field, UCB$L_TU_RECORD in favor of
0000    ;               the device independent field, UCB$L_RECORD.
0000    ;
0000    ;       V03-123 ROW0253         Ralph O. Weber          12-NOV-1983
0000    ;               Change device dependent UCB definitions to work with globally
0000    ;               defined MSCP extension to the UCB.  This change does not make
0000    ;               use of all the UCB fields in the new extension.  It simply
0000    ;               eliminates interactions which will prevent this module from
0000    ;               building in the presence of the new UCB definitions.  The
```

```
0000  ;                 UCB$L_TU_MEDIATYP field, which was changed to UCB$L_MEDIA_ID
0000  ;                 ages ago, has also been eliminated.  NB: a gross hack has been
0000  ;                 employed to keep this driver compatible with the other magtape
0000  ;                 drivers and the magtape ACP.  This will be corrected when all
0000  ;                 the involved parties start using the newly defined
0000  ;                 UCB$L_RECORD.
0000  ;
0000  ;         V03-122 ROW0245           Ralph O. Weber          19-OCT-1983
0000  ;                 Correct couple of outstanding bugs:
0000  ;                  - Change TU$IDR to store incomming message size in
0000  ;                    CDRP$L_IOST2+2.  This provides the message size to any code
0000  ;                    requiring it.  In particular, the INVALID_STS fixes
0000  ;                    mentioned below use this feature.
0000  ;                  - Fix INVALID_STS to properly place the size of the incomming
0000  ;                    MSCP message in R1 before calling ERL$LOG_DMSCP.
0000  ;
0000  ;         V03-121 ROW0243           Ralph O. Weber          17-OCT-1983
0000  ;                 Enhance SEQ_ENDCHECK to allow canceled (MSCP aborted) end
0000  ;                 packets to be received out of sequence.  This produces
0000  ;                 conformance to a revised version of the TMSCP specification.
0000  ;
0000  ;         V03-120 ROW0242           Ralph O. Weber          17-OCT-1983
0000  ;                 Change unit attention processing in DU$IDR to skip altering
0000  ;                 UCB$M_DU_WAITBMP and UCB$W_RWAITCNT when the CDDB$M_INITING or
0000  ;                 CDDB$M_RECONNECT is set in CDDB$W_STATUS.  This prevents
0000  ;                 altering the wait count is such a way that the wait count
0000  ;                 tests in controller init and reconnection processing fail.
0000  ;                 Therefore, a spurous disk class driver bugcheck is eliminated.
0000  ;
0000  ;         V03-119 BLS0234           Benn Schreiber          9-Aug-1983
0000  ;                 Add missing G^s to calls in exec.
0000  ;
0000  ;         V03-118 RLRDLATE          Robert L. Rappaport     25-Jul-1983
0000  ;                 Check for Data Late subcode in Controller Errors on
0000  ;                 data transfer commands, and return SS$_DATALATE.
0000  ;
0000  ;         V03-117 RLRDLEOT          Robert L. Rappaport     19-Jul-1983
0000  ;                 Implement suport for new MSCP$M_MD_DLEOT modifier.
0000  ;                 Modifier means "Detect Logical End Of Tape" and is
0000  ;                 used on QIO Skip files and Skip records (forward
0000  ;                 direction only).
0000  ;
0000  ;         V03-116 RLRIMMED          Robert L. Rappaport     19-Jul-1983
0000  ;                 Implement support for new MSCP$M_MD_IMMED modifier
0000  ;                 that allows us to express that certain commands,
0000  ;                 namely REWIND and DSE, are to return their End Messages
0000  ;                 when the command BEGINS to execute rather than when it
0000  ;                 completes.  A discussion of this is found in the TMSCP
0000  ;                 spec under "Synchronous versus Asynchronous" operation
0000  ;                 of lengthy commands.
0000  ;
0000  ;                 The effort here consists of simplifying greatly the
0000  ;                 previous method of implementing support for IO$M_NOWAIT.
0000  ;                 This simplification eliminates the need for a REWIND
0000  ;                 CDRP, as well as the need for special handling of
0000  ;                 Rewind and Available (UNLOAD) requests.
0000  ;
```

```
0000    ;               This update almost completely obviates those changes
0000    ;               implemented as a result of update RLRRWATN.
0000    ;
0000    ;               Also in this update fix bug in START_SETCHAR wherein we
0000    ;               neglected to call SCS$UNSTALLUCB after decrementing
0000    ;               UCB$W_RWAITCNT.
0000    ;
0000    ;       V03-115 RLRUPTODATE     Robert L. Rappaport     26-Jul-1983
0000    ;               Adapt and incorporate relevant changes from Disk
0000    ;               Class Driver.  From ;RLRDDB audit of DUDRIVER
0000    ;               thru ;RLRODDBCNT.
0000    ;
0000    ;       V03-114 RLRGROWTH       Robert L. Rappaport     23-Jun-1983
0000    ;               Due to growth in the CDDB, the length of the CDDB plus
0000    ;               the length of the CDRP is NOT < 256.  We must change
0000    ;               a MOVZBL to a MOVZWL.
0000    ;
0000    ;       V03-113 RLRDPATH2       Robert L. Rappaport     31-May-1983
0000    ;               As a result of the previous change (RLRDPATH1),
0000    ;               UCB$L_TU_RECORD has moved with respect to UCB$L_DPC
0000    ;               breaking an assume statement that must now be fixed.
0000    ;
0000    ;       V03-112 RLRDPATH1       Robert L. Rappaport     25-May-1983
0000    ;               Allow UCB to include new DUAL PORT extension by
0000    ;               changing base of where we begin the private TUDRIVER
0000    ;               extension from UCB$L_DPC+4 to UCB$L_DP_LINK+4.
0000    ;
0000    ;       V03-111 RLRRWCPTRa      Robert L. Rappaport     11-Apr-1983
0000    ;               Correct bug in RLRRWCPTR fix.
0000    ;
0000    ;       V03-110 RLRCANCELf      Robert L. Rappaport     11-Apr-1983
0000    ;               Initialize CDRP fields before deciding whether to start
0000    ;               this I/O request or whether to Q to UCB I/O Queue.  This
0000    ;               prevents misinterpreting uninitialized fields.
0000    ;
0000    ;       V03-109 RLRRWCPTR       Robert L. Rappaport     4-Mar-1983
0000    ;               Test for zero UCB$L_RWCPTR in RDTWAIT_DIS_ACT and
0000    ;               in RDT_DIS_ACTION.  Such a situation could occur if
0000    ;               no RSPID's were available during a re-Connection and
0000    ;               if the re-Connection failed and we had to do a
0000    ;               re-re-Connection.  Also use Controller timeout for
0000    ;               host timeout value for those controllers for which
0000    ;               we care to set a host timeout.  Also only use INIT_IMMED_DELTA
0000    ;               for timing out the first SET_CONTROLLER_CHAR command. After-
0000    ;               words always use CDDB$W_CNTRLTMO.  Also increase
0000    ;               INIT_IMMED_DELTA to 30.
0000    ;
0000    ;       V03-108 RLRTMUCB        Robert L. Rappaport     25-Feb-1983
0000    ;               Revamp Template UCB so as to be automatically compliant
0000    ;               with new UCB additions.  Also remove initial Breakpoint.
0000    ;
0000    ;       V03-107 RLRWTMPOS       Robert L. Rappaport     22-Feb-1983
0000    ;               Update UCB$L_TU_POSITION after error on WRITE TAPE MARK
0000    ;               command.
0000    ;
0000    ;       V03-106 RLRSEQNOP       Robert L. Rappaport     15-Feb-1983
0000    ;               Use REPOSITION command with zeroes as a sequential NOP
```

```
0000    ;               in SET CHAR and SET MODE processing.
0000    ;
0000    ;       V03-105 RLRWRTM         Robert L. Rappaport     14-Feb-1983
0000    ;               Accept MSCP$K_ST_DATA as possible status of Write Tape Mark.
0000    ;
0000    ;       V03-104 RLRRWATN        Robert L. Rappaport     11-Feb-1983
0000    ;               Implement REWIND ATTENTION and NOWAIT.  Also add
0000    ;               support for REWIND Attention messages received as a
0000    ;               AVAILABLE and UNLOAD commands.  Also support ignoring
0000    ;               of spurious REWIND Attention messages.
0000    ;
0000    ;       V03-103 RLRTRACE        Robert L. Rappaport     4-Feb-1983
0000    ;               Make IRP trace a per unit rather than a per system
0000    ;               structure by moving it to the UCB.
0000    ;
0000    ; MACRO LIBRARY CALLS
0000    ;
0000
0000            $CDDBDEF                        ;Define CDDB offsets
0000            $CDRPDEF                        ;Define CDRP offsets
0000            $CDTDEF                         ;Define CDT offsets
0000            $CRBDEF                         ;Define CRB offsets
0000            $DCDEF                          ;Define Device Classes and Types
0000            $DDBDEF                         ;Define DDB offsets
0000            $DEVDEF                         ;Define DEVICE CHARACTERISTICS bits
0000            $DPTDEF                         ;Define DPT offsets
0000            $DYNDEF                         ;Define DYN symbols
0000            $EMBLTDEF                       ;Define EMB Log Message Types
0000            $FKBDEF                         ;Define FKB offsets
0000            $IDBDEF                         ;Define IDB offsets
0000            $IODEF                          ;Define I/O FUNCTION codes
0000            $IPLDEF                         ;Define symbolic IPL's
0000            $IRPDEF                         ;Define IRP offsets
0000            $MSCPDEF                        ;Define MSCP packet offsets
0000            $MSLGDEF                        ;Define MSCP Error Log offsets
0000            $MTDEF                          ;Define MAGTAPE STATUS bits
0000            $ORBDEF                         ;Define ORB offsets
0000            $PBDEF                          ;Define Path Block offsets
0000            $PCBDEF                         ;Define PCB offsets
0000            $PDTDEF                         ;Define PDT offsets
0000            $PRDEF                          ;Define Processor Registers
0000            $SBDEF                          ;Define System Block Offsets
0000            $SCSCMGDEF                      ;Define SCS Connect Message offsets
0000            $RCTDEF                         ;Define RCT offsets
0000            $RDDEF                          ;Define RDTE offsets
0000            $RDTDEF                         ;Define RDT offsets
0000            $SSDEF                          ;Define System Status values
0000            $UCBDEF                         ;Define UCB offsets
0000            $VADEF                          ;Define Virtual Address offsets
0000            $VECDEF                         ;Define INTERRUPT DISPATCH VECTOR offsets
0000            $WCBDEF                         ;Define WCB offsets
0000
0000
0000            $DUTUDEF                        ;Define common class driver CDDB
0000                                            ; extensions and other common symbols
0000
0000
```

```
                    0000     457 ; Constants
                    0000     458
          00000001  0000     459 ALLOC_DELTA=1                     ; Number of seconds to wait to retry pool
                    0000     460                                   ;  allocation that failed.
          0000001E  0000     461 INIT_IMMED_DELTA=30               ; During Controller Initialization, the
                    0000     462                                   ;  timeout DELTA for immediate MSCP commands.
          0000000A  0000     463 CONNECT_DELTA=10                  ; During Controller Initialization, the
                    0000     464                                   ;  time interval for retrying failed
                    0000     465                                   ;  CONNECT attempts.
          0000001E  0000     466 HOST_TIMEOUT=30                   ; Host timeout value.
                    0000     467
          00000001  0000     468 DISCONNECT_REASON=1
          0000000A  0000     469 INITIAL_CREDIT=10
          00000002  0000     470 INITIAL_DG_COUNT=2
          00000002  0000     471 MAX_RETRY=2
          00000002  0000     472 MIN_SEND_CREDIT=2
```

```
0000            .SBTTL  MACRO DEFINITIONS
0000
0000    ;
0000    ; Expanded opcode macros - Branch word conditional psuedo opcodes.
0000    ;
0000
0000    ;
0000    ; BWNEQ - Branch (word offset) not equal
0000    ;
0000
0000            .MACRO  BWNEQ   DEST,?L1
0000            BEQL    L1                      ; Branch around if NOT NEQ.
0000            BRW     DEST                    ; Branch to destination if NEQ.
0000    L1:                                     ; Around.
0000            .ENDM   BWNEQ
0000
0000
0000    ;
0000    ; BWEQL - Branch (word offset) equal
0000    ;
0000
0000            .MACRO  BWEQL   DEST,?L1
0000            .SHOW
0000            BNEQ    L1                      ; Branch around if NOT EQL.
0000            BRW     DEST                    ; Branch to destination if EQL.
0000    L1:                                     ; Around.
0000            .NOSHOW
0000            .ENDM   BWEQL
0000
0000    ;
0000    ; BWBS - Branch (word offset) bit set.
0000    ;
0000
0000            .MACRO  BWBS    BIT,FIELD,DEST,?L1
0000            .SHOW
0000            BBC     BIT,FIELD,L1                    ; Branch around if bit NOT set.
0000            BRW     DEST                            ; Branch to destination if bit set.
0000    L1:                                             ; Around.
0000            .NOSHOW
0000            .ENDM   BWBS
0000
0000    ;
0000    ; BWBC - Branch (word offset) bit clear.
0000    ;
0000
0000            .MACRO  BWBC    BIT,FIELD,DEST,?L1
0000            .SHOW
0000            BBS     BIT,FIELD,L1                    ; Branch around if bit NOT clear.
0000            BRW     DEST                            ; Branch to destination if bit clear.
0000    L1:                                             ; Around.
0000            .NOSHOW
0000            .ENDM   BWBC
0000
0000            .IF     DF      TU_SEQCHK
0000    ;
0000    ; SEQFUNC - Macro included in conditional code to check sequentiality
0000    ;               of function terminations.
```

```
0000       ;
0000
0000               .MACRO  SEQFUNC CODES
0000       MASKL   = 0
0000       MASKH   = 0
0000               .IRP    X,<CODES>
0000               .IF     GT      <IO$_'X&IO$_VIRTUAL>-31
0000       MASKH   = MASKH!<1@<<IO$_'X&IO$_VIRTUAL>-32>>
0000               .IFF
0000       MASKL   = MASKL!<1@<IO$_'X&IO$_VIRTUAL>>
0000               .ENDC
0000               .ENDM
0000               .LONG   MASKL,MASKH
0000               .ENDM   SEQFUNC
0000               .ENDC
0000
0000       ;
0000       ; START_SEQNOP - macro to start a sequential NOP sequence
0000       ;
0000       ;       This macro starts a sequential NOP sequence.  A sequential NOP
0000       ;       sequence encapsulates a series of TMSCP operations which must occur
0000       ;       sequentially with respect to the stream of TMSCP operations flowing
0000       ;       through the driver.
0000       ;
0000       ;       First UCB$W_RWAITCNT is increased by one to prevent future I/O
0000       ;       requests from starting.  Then a TMSCP sequential command which does
0000       ;       not alter the tape position is sent to the server.  When the
0000       ;       sequential command completes, the driver and the server are
0000       ;       synchronized.
0000       ;
0000       ;       Upon exit from this macro, the currently executing thread is the only
0000       ;       thread conversing with the server.  When the operations which must be
0000       ;       done in this sychronized state are completed, the sequential NOP state
0000       ;       should be terminated using the END_SEQNOP macro.
0000       ;
0000       ; Inputs:
0000       ;
0000       ;       R3      UCB address
0000       ;       R4      PDT address
0000       ;       R5      CDRP address (RSPID & message buffer already allocated and
0000       ;                               initialized)
0000       ;       (SP)    address of caller's caller
0000       ;
0000       ; Outputs:
0000       ;
0000       ;       R3 through R5 unchanged
0000       ;       All other registers altered
0000
0000               .MACRO  START_SEQNOP ?L1
0000               BBSS    #UCB$V_TU_SEQNOP, -     ; Set sequential NOP in progress and
0000                       UCB$W_DEVSTS(R3), L1    ; branch if its already set.
0000               INCW    UCB$W_RWAITCNT(R3)      ; Else, increment wait count to
0000                                               ; disallow I/O.
0000       L1:     MOVB    #MSCP$K_OP_REPOS,-      ; Transfer REPOSITION opcode
0000                       MSCP$B_OPCODE(R2)       ;  to packet.
0000               ASSUME  MSCP$V_MD_CLSEX GE 8
0000               BICB    #<MSCP$M_MD_CLSEX@-8>,- ; Specifically never clear SEX on the
```

```
0000                    MSCP$W_MODIFIER+1(R2)   ;  Seq. NOP command of a SETMODE.
0000            SEND_MSCP_MSG                   ; Send message to remote MSCP server.
0000            RESET_MSCP_MSG                  ; Setup message buf. etc. for reuse.
0000                                            ;  refresh RSPID, MSG_BUF, etc.
0000            .ENDM   START_SEQNOP
0000
0000    ;
0000    ; END_SEQNOP - terminate sequential NOP sequence
0000    ;
0000    ;       This macro terminates the class driver - server synchronization
0000    ;       established by START_SEQNOP and returns the communications to a full
0000    ;       stream ahead mode.
0000    ;
0000    ; Inputs:
0000    ;
0000    ;       R3      UCB address
0000    ;
0000    ; Outputs:
0000    ;
0000    ;       R0 and R3 through R5 unchanged
0000    ;       All other registers altered
0000
0000            .MACRO  END_SEQNOP ?END
0000            BICW    #UCB$M_TU_SEQNOP, -     ; Indicate sequential NOP is no longer
0000                    UCB$W_DEVSTS(R3)        ; in progress.
0000            DECW    UCB$W_RWAITCNT(R3)      ; Decrement wait count to allow I/O.
0000            BNEQ    END                     ; Branch if wait count not zero.
0000            PUSHR   #^M<R0,R3,R4,R5>        ; Save valuable registers.
0000            MOVL    R3, R5                  ; R5 => UCB for SCS$UNSTALLUCB.
0000            JSB     G^SCS$UNSTALLUCB        ; Start up any waiting IRPs on this UCB.
0000            POPR    #^M<R0,R3,R4,R5>        ; Restore valuable registers.
0000    END:
0000            .ENDM   END_SEQNOP
```

```
0000          .SBTTL  ASSUMES
0000
0000  ; The following set of ASSUME statements will all be true as long as
0000  ;       the IRP and CDRP definitions remain consistent.
0000
0000          ASSUME  CDRP$L_IOQFL-CDRP$L_IOQFL          EQ      IRP$L_IOQFL
0000          ASSUME  CDRP$L_IOQBL-CDRP$L_IOQFL          EQ      IRP$L_IOQBL
0000          ASSUME  CDRP$W_IRP_SIZE-CDRP$L_IOQFL       EQ      IRP$W_SIZE
0000          ASSUME  CDRP$B_IRP_TYPE-CDRP$L_IOQFL       EQ      IRP$B_TYPE
0000          ASSUME  CDRP$B_RMOD-CDRP$L_IOQFL           EQ      IRP$B_RMOD
0000          ASSUME  CDRP$L_PID-CDRP$L_IOQFL            EQ      IRP$L_PID
0000          ASSUME  CDRP$L_AST-CDRP$L_IOQFL            EQ      IRP$L_AST
0000          ASSUME  CDRP$L_ASTPRM-CDRP$L_IOQFL         EQ      IRP$L_ASTPRM
0000          ASSUME  CDRP$L_WIND-CDRP$L_IOQFL           EQ      IRP$L_WIND
0000          ASSUME  CDRP$L_UCB-CDRP$L_IOQFL            EQ      IRP$L_UCB
0000          ASSUME  CDRP$W_FUNC-CDRP$L_IOQFL           EQ      IRP$W_FUNC
0000          ASSUME  CDRP$B_EFN-CDRP$L_IOQFL            EQ      IRP$B_EFN
0000          ASSUME  CDRP$B_PRI-CDRP$L_IOQFL            EQ      IRP$B_PRI
0000          ASSUME  CDRP$L_IOSB-CDRP$L_IOQFL           EQ      IRP$L_IOSB
0000          ASSUME  CDRP$W_CHAN-CDRP$L_IOQFL           EQ      IRP$W_CHAN
0000          ASSUME  CDRP$W_STS-CDRP$L_IOQFL            EQ      IRP$W_STS
0000          ASSUME  CDRP$L_SVAPTE-CDRP$L_IOQFL         EQ      IRP$L_SVAPTE
0000          ASSUME  CDRP$W_BOFF-CDRP$L_IOQFL           EQ      IRP$W_BOFF
0000          ASSUME  CDRP$L_BCNT-CDRP$L_IOQFL           EQ      IRP$L_BCNT
0000          ASSUME  CDRP$W_BCNT-CDRP$L_IOQFL           EQ      IRP$W_BCNT
0000          ASSUME  CDRP$L_IOST1-CDRP$L_IOQFL          EQ      IRP$L_IOST1
0000          ASSUME  CDRP$L_MEDIA-CDRP$L_IOQFL          EQ      IRP$L_MEDIA
0000          ASSUME  CDRP$L_IOST2-CDRP$L_IOQFL          EQ      IRP$L_IOST2
0000          ASSUME  CDRP$L_TT_TERM-CDRP$L_IOQFL        EQ      IRP$L_TT_TERM
0000          ASSUME  CDRP$B_CARCON-CDRP$L_IOQFL         EQ      IRP$B_CARCON
0000          ASSUME  CDRP$Q_NT_PRVMSK-CDRP$L_IOQFL      EQ      IRP$Q_NT_PRVMSK
0000          ASSUME  CDRP$L_ABCNT-CDRP$L_IOQFL          EQ      IRP$L_ABCNT
0000          ASSUME  CDRP$W_ABCNT-CDRP$L_IOQFL          EQ      IRP$W_ABCNT
0000          ASSUME  CDRP$L_OBCNT-CDRP$L_IOQFL          EQ      IRP$L_OBCNT
0000          ASSUME  CDRP$W_OBCNT-CDRP$L_IOQFL          EQ      IRP$W_OBCNT
0000          ASSUME  CDRP$L_SEGVBN-CDRP$L_IOQFL         EQ      IRP$L_SEGVBN
0000          ASSUME  CDRP$L_JNL_SEQNO-CDRP$L_IOQFL      EQ      IRP$L_JNL_SEQNO
0000          ASSUME  CDRP$L_DIAGBUF-CDRP$L_IOQFL        EQ      IRP$L_DIAGBUF
0000          ASSUME  CDRP$L_SEQNUM-CDRP$L_IOQFL         EQ      IRP$L_SEQNUM
0000          ASSUME  CDRP$L_EXTEND-CDRP$L_IOQFL         EQ      IRP$L_EXTEND
0000          ASSUME  CDRP$L_ARB-CDRP$L_IOQFL            EQ      IRP$L_ARB
```

```
                 0000           .SBTTL  TAPE CLASS DRIVER DEVICE DEPENDENT UNIT CONTROL BLOCK OFFSETS
                 0000
                 0000           $DEFINI UCB
                 0000
                 0000
        000000EC 0000    .=UCB$K_MSCP_TAPE_LENGTH
                 00EC
                 00EC    $DEF   UCB$L_TU_MAXWRCNT                     ; Largest size record likely to have
        000000F0 00EC                           .BLKL    1            ;  reliability statistics.
                 00F0    $DEF   UCB$W_TU_FORMAT .BLKW    1            ; Format (density).
                 00F2    $DEF   UCB$W_TU_SPEED  .BLKW    1            ; Current speed.
                 00F4    $DEF   UCB$W_TU_NOISE  .BLKW    1            ; Size of noise records ignored by
                 00F6                                                 ;  controller.
                 00F6           .IF     DF      TU_SEQCHK
                 00F6    $DEF   UCB$B_TU_OLDINX .BLKB    1            ; Index of oldest Sequence number.
                 00F6    $DEF   UCB$B_TU_NEWINX .BLKB    1            ; Index of nexat available Seq. # slot.
                 00F6    $DEF   UCB$L_TU_SEQARY .BLKL    64           ; Array of 64 longwords wherein we
                 00F6                                                 ;  we save IRP sequence numbers.
                 00F6           .IFF
        000000F8 00F6                           .BLKW    1            ; Reserved.
                 00F8           .ENDC
                 00F8
                 00F8           .IF     DF      TU_TRACE
                 00F8    $DEF   UCB$L_TRACEBEG  .BLKL    1            ; Pointer to beginning of trace ring.
                 00F8    $DEF   UCB$L_TRACEPTR  .BLKL    1            ; Pointer to next available slot.
                 00F8    $DEF   UCB$L_TRACEND   .BLKL    1            ; Pointer to beyond trace ring.
                 00F8
                 00F8           .ENDC
                 00F8
        000000F8 00F8    UCB$K_TU_LENGTH=.
                 00F8
                 00F8           $DEFEND UCB
                 0000
                 0000
                 0000           .SBTTL  Allocate Space for Template UCB
                 0000
                 0000    ; Allocate zeroed space for template UCB.
                 0000
                 0000           INIT_UCB size=UCB$K_TU_LENGTH
                 0000           INIT_ORB size=ORB$C_LENGTH
```

```
0000            .SBTTL  DRIVER PROLOGUE AND DISPATCH TABLES (and UCB Initialization)
0000    ;
0000    ; LOCAL DATA
0000    ;
0000    ; DRIVER PROLOGUE TABLE
0000    ;
0000
0000            DPTAB   -                       ;DEFINE DRIVER PROLOGUE TABLE
0000                    END=DUTU$END,-          ; End of driver
0000                    ADAPTER=NULL,-          ; No Adapter
0000                    FLAGS=<DPT$M_SCS -      ; Driver requires that SCS be loaded
0000                            !DPT$M_NOUNLOAD>,-; Driver cannot be reloaded
0000                    UCBSIZE=UCB$K_TU_LENGTH,-;Sysgen insists on making a UCB
0000                    MAXUNITS=1,-            ;Sysgen insists on making a UCB
0000                    NAME=TUDRIVER                   ; Driver name
0038            DPT_STORE INIT                          ; Control block init values
0038            DPT_STORE DDB,DDB$L_ACPD,L,<^A\MTA\>    ; Default ACP name
003F
003F
003F            ; The following UCB initialization requests alter the template UCB
003F            ; as well as producing equivalent DPT_STORE entries.  Thus both
003F            ; structures reflect the required initial UCB state and the UCBs
003F            ; initially processed by this driver are identical whether they are
003F            ; produced by SYSGEN or by IOC$COPY_UCB.
003F
003F            INIT_UCB        W_SIZE,WORD,UCB$K_TU_LENGTH
003F            INIT_UCB        B_TYPE,BYTE,DYN$C_UCB
003F            INIT_UCB        B_FIPL,BYTE,IPL$_SCS
0043            INIT_UCB        L_DEVCHAR,LONG,<<DEV$M_FOD!-
0043                                             DEV$M_DIR!-
0043                                             DEV$M_AVL!-
0043                                             DEV$M_ELG!-
0043                                             DEV$M_IDV!-
0043                                             DEV$M_ODV!-
0043                                             DEV$M_SDI!-
0043                                             DEV$M_SQD>>
004A            INIT_UCB        L_DEVCHAR2,LONG,<<DEV$M_CLU!-
004A                                              DEV$M_MSCP!-
004A                                              DEV$M_NNM>>
0051            INIT_UCB        B_DEVCLASS,BYTE,DC$_TAPE
0055            INIT_UCB        W_DEVBUFSIZ,WORD,2048
005A            INIT_UCB        L_DEVDEPEND,LONG,<<<MT$K_NORMAL11 @ MT$V_FORMAT>!-
005A                                               <MT$K_PE_1600 @ MT$V_DENSITY>>>
0061            INIT_UCB        W_RWAITCNT,WORD,1
0066            INIT_UCB        B_DIPL,BYTE,IPL$_SCS
006A            INIT_UCB        W_DEVSTS,WORD, <<UCB$M_MSCP_INITING -
006A                                            !UCB$M_MSCP_WAITBMP>>
006F
006F            ; The following ORB initialization requests alter the template ORB
006F            ; as well as producing equivalent DPT_STORE entries.  Thus both
006F            ; structures reflect the required initial ORB state and the ORBs
006F            ; initially processed by this driver are identical whether they are
006F            ; produced by SYSGEN or by IOC$COPY_UCB.
006F
006F            INIT_ORB        W_SIZE,WORD,ORB$C_LENGTH
006F            INIT_ORB        B_TYPE,BYTE,DYN$C_ORB
006F            INIT_ORB        B_FLAGS,BYTE,<< -
```

```
006F                            ORB$M_PROT_16>>         ; SOGW protection word
0073            INIT_ORB        W_PROT,WORD,0           ; default protection
0078            INIT_ORB        L_OWNER,LONG,0          ; no owner as yet
0078            DPT_STORE REINIT                        ; Control block re-initialization values
0078
0078            ; N.B. Causing the following values to be setup during re-initializa-
0078            ; tion is not significant because this driver cannot be reloaded.
0078            ; However, were the driver to be reloadable the following values would
0078            ; need to be re-initialized upon each driver reload.
0078
0078            DPT_STORE CRB, -                                ; Controller init routine.
0078                    CRB$L_INTD+VEC$L_INITIAL,D,TU_CONTROLLER_INIT
007D            DPT_STORE DDB,DDB$L_DDT,D,TU$DDT                ; DDT address.
0082
0082            DPT_STORE END
0000
0000    ;
0000    ; DRIVER DISPATCH TABLE
0000    ;
0000
0000            DDTAB   DEVNAM=TU,-                     ;DRIVER DISPATCH TABLE
0000                    START=TU_STARTIO,-              ;START I/O OPERATION
0000                    UNSOLIC=TU_UNSOLNT,-            ;UNSOLICITED INTERRUPT
0000                    FUNCTB=TU_FUNCTABLE,-           ;FUNCTION DECISION TABLE
0000                    CANCEL=DUTU$CANCEL, -           ;CANCEL I/O ENTRY POINT
0000                    REGDMP=0,-                      ;REGISTER DUMP ROUTINE
0000                    DIAGBF=MSCP$K_MXCMDLEN+MSCP$K_LEN+20+12,-; DIAG BUFF SIZE
0000                    ERLGBF=0,-                      ; ERLG BUFF SIZE
0000                    UNITINIT=DUTU$UNITINIT,-        ; Unit initialization routine.
0000                    ALTSTART=0                      ; Alternate Start I/O entry.
```

```
0038         .SBTTL  DISK CLASS DRIVER FUNCTION DECISION TABLE
0038  ;+
0038  ; TAPE CLASS DRIVER FUNLTION DECISION TABLE
0038  ;-
0038
0038  TU_FUNCTABLE:                                       ;Function Decision Table
0038          FUNCTAB ,-                                  ; LEGAL FUNCTIONS
0038                  <NOP,-                              ;  No operation
0038                   UNLOAD,-                           ;  Unload (make available + spindown)
0038                   AVAILABLE,-                        ;  Available (no spindown)
0038                   SPACERECORD,-                      ;  Space Records
0038                   RECAL,-                            ;  Recalibrate (REWIND)
0038                   PACKACK,-                          ;  Pack Acknowledge
0038                   ERASETAPE,-                        ;  Erase Tape (Erase Gap)
0038                   SENSECHAR,-                        ;  Sense Characteristics
0038                   SETCHAR,-                          ;  Set Characteristics
0038                   SENSEMODE,-                        ;  Sense Mode
0038                   SETMODE,-                          ;  Set Mode
0038                   SPACEFILE,-                        ;  Space File
0038                   WRITECHECK,-                       ;  Write Check
0038                   READPBLK,-                         ;  Read  PHYSICAL Block
0038                   WRITEPBLK,-                        ;  Write PHYSICAL Block
0038                   READLBLK,-                         ;  Read  LOGICAL  Block
0038                   WRITELBLK,-                        ;  Write LOGICAL  Block
0038                   READVBLK,-                         ;  Read  VIRTUAL  Block
0038                   WRITEVBLK,-                        ;  Write VIRTUAL  Block
0038                   WRITEMARK,-                        ;  Write Tape Mark
0038                   DSE,-                              ;  Data Security Erase
0038                   REWIND,-                           ;  Rewind
0038                   REWINDOFF,-                        ;  Rewind AND Set Offline (UNLOAD)
0038                   SKIPRECORD,-                       ;  Skip Records
0038                   SKIPFILE,-                         ;  Skip Files
0038                   WRITEOF,-                          ;  Write End Of File
0038                   ACCESS,-                           ;  Access file and/or find directory entry
0038                   ACPCONTROL,-                       ;  ACP Control Function
0038                   CREATE,-                           ;  Create file and/or create directory entry
0038                   DEACCESS,-                         ;  Deaccess file
0038                   DELETE,-                           ;  Delete file and/or directory entry
0038                   MODIFY,-                           ;  Modify file attributes
0038                   MOUNT>                             ;  Mount volume
0040          FUNCTAB ,-                                  ; BUFFERED I/O FUNCTIONS
0040                  <NOP,-                              ;  No Operation
0040                   UNLOAD,-                           ;  Unload (make available + spindown)
0040                   AVAILABLE,-                        ;  Available (no spindown)
0040                   SPACERECORD,-                      ;  Space Records
0040                   RECAL,-                            ;  Recalibrate (REWIND)
0040                   PACKACK,-                          ;  Pack Acknowledge
0040                   ERASETAPE,-                        ;  Erase Tape (Erase Gap)
0040                   SENSECHAR,-                        ;  Sense Characteristics
0040                   SETCHAR,-                          ;  Set   Characteristics
0040                   SENSEMODE,-                        ;  Sense Mode
0040                   SETMODE,-                          ;  Set   Mode
0040                   SPACEFILE,-                        ;  Space File
0040                   WRITEMARK,-                        ;  Write Tape Mark
0040                   DSE,-                              ;  Data Security Erase
0040                   REWIND,-                           ;  Rewind
0040                   REWINDOFF,-                        ;  Rewind AND Set Offline (UNLOAD)
```

```
            0040    850                      SKIPRECORD,-              ;  Skip Records
            0040    851                      SKIPFILE,-                ;  Skip Files
            0040    852                      WRITEOF,-                 ;  Write End Of File
            0040    853                      ACCESS,-                  ;  Access file and/or find directory entry
            0040    854                      ACPCONTROL,-              ;  ACP Control Function
            0040    855                      CREATE,-                  ;  Create file and/or create directory entry
            0040    856                      DEACCESS,-                ;  Deaccess file
            0040    857                      DELETE,-                  ;  Delete file and/or directory entry
            0040    858                      MODIFY,-                  ;  Modify file attributes
            0040    859                      MOUNT>                    ;  Mount volume
            0048    860             FUNCTAB +ACP$READBLK,-             ; READ FUNCTIONS
            0048    861                     <READLBLK,-                ;  Read LOGICAL Block
            0048    862                      READPBLK,-                ;  Read  PHYSICAL Block
            0048    863                      READVBLK>                 ;  Read VIRTUAL Block
            0054    864             FUNCTAB +ACP$WRITEBLK,-            ; WRITE FUNCTIONS
            0054    865                     <WRITECHECK,-              ;  Write Check
            0054    866                      WRITEPBLK,-               ;  Write PHYSICAL Block
            0054    867                      WRITELBLK,-               ;  Write LOGICAL Block
            0054    868                      WRITEVBLK>                ;  Write VIRTUAL Block
            0060    869             FUNCTAB +ACP$ACCESS,-              ;
            0060    870                     <ACCESS,CREATE>            ; ACCESS AND CREATE FILE OR DIRECTORY
            006C    871             FUNCTAB +ACP$DEACCESS,<DEACCESS>;  DEACCESS FILE
            0078    872             FUNCTAB +ACP$MODIFY,-              ;
            0078    873                     <ACPCONTROL,-              ;  ACP Control Function
            0078    874                      DELETE,-                  ;  Delete file or directory entry
            0078    875                      MODIFY>                   ;  Modify File Attributes
            0084    876             FUNCTAB +ACP$MOUNT,<MOUNT>         ; Mount Volume
            0090    877             FUNCTAB +MT$CHECK_ACCESS,-         ;MAGTAPE CHECK ACCESS FUNCTIONS
            0090    878                     <ERASETAPE,-               ;  Erase Tape (Erase Gap)
            0090    879                      WRITEMARK,-               ;  Write Tape Mark
            0090    880                      DSE,-                     ;  Data Security Erase
            0090    881                      WRITEOF>                  ;  Write End Of File
            009C    882             FUNCTAB +EXE$ZEROPARM,-            ; ZERO PARAMETER FUNCTIONS
            009C    883                     <NOP,-                     ;  No Operation
            009C    884                      UNLOAD,-                  ;  Unload (make available + spindown)
            009C    885                      RECAL,-                   ;  Recalibrate (REWIND)
            009C    886                      REWIND,-                  ;  Rewind
            009C    887                      REWINDOFF,-               ;  Rewind AND Set Offline (UNLOAD)
            009C    888                      ERASETAPE,-               ;  Erase Tape (Erase Gap)
            009C    889                      SENSECHAR,-               ;  Sense Characteristics
            009C    890                      SENSEMODE,-               ;  Sense Mode
            009C    891                      WRITEMARK,-               ;  Write Tape Mark
            009C    892                      DSE,-                     ;  Data Security Erase
            009C    893                      WRITEOF,-                 ;  Write End Of File
            009C    894                      AVAILABLE,-               ;  Available (no spindown)
            009C    895                      PACKACK>                  ;  Pack Acknowledge
            00A8    896             FUNCTAB +EXE$ONEPARM,-             ;ONE PARAMETER FUNCTIONS
            00A8    897                     <SPACERECORD,-             ;  Space Records
            00A8    898                      SPACEFILE,-               ;  Space Files
            00A8    899                      SKIPRECORD,-              ;  Skip Records
            00A8    900                      SKIPFILE>                 ;  Skip Files
            00B4    901             FUNCTAB +EXE$SETMODE,-             ;SET TAPE CHARACTERISTICS
            00B4    902                     <SETCHAR,-                 ;
            00B4    903                      SETMODE>                  ;
```

```
                00C0          .SBTTL  Static Storage
                00C0          .SBTTL  -       Data Area Shared With Common Subroutines Module
                00C0    ;++
                00C0    ;
                00C0    ; Data Area Shared With Common Subroutines Module
                00C0    ;
                00C0    ; Functional Description:
                00C0    ;
                00C0    ;       This PSECT contains those constant (link-time) values which would
                00C0    ;       otherwise be passed as arguments to the disk and tape class driver
                00C0    ;       common routines in module DUTUSUBS.
                00C0    ;
                00C0    ;--
                00C0
                00C0          .SAVE
                00C0
            00000000          .PSECT  $$$220_DUTU_DATA_01 RD,WRT,EXE,LONG
                0000
                0000          ASSUME  DUTU$L_CDDB_LISTHEAD EQ 0
                0000
                0000    ;base + DUTU$L_CDDB_LISTHEAD                    ; Location containing the
                0000                                                    ; address of the CDDB listhead
      00000000' 0000          .ADDRESS IOC$GL_TU_CDDB                   ; for CDDBs belonging to the
                0004                                                    ; tape device type
                0004
            000000C0          .RESTORE
```

```
                  00C0   932           .SBTTL  -       Media-id to Device Type Conversion Table
                  00C0   933 ;++
                  00C0   934 ;
                  00C0   935 ; Media-id to Device Type Conversion Table
                  00C0   936 ;
                  00C0   937 ; Functional Description:
                  00C0   938 ;
                  00C0   939 ;         This table is used by DUTU$GET_DEVTYPE to convert a MSCP media
                  00C0   940 ;         identifier to a VMS device type.
                  00C0   941 ;
                  00C0   942 ;         Entries are made here in order of expected frequency of use.  This
                  00C0   943 ;         speeds lookup for the more common cases.
                  00C0   944 ;
                  00C0   945 ;--
                  00C0   946
                  00C0   947           MEDIA   <MU>, <TU81>
                  0000
        6D695051  0000                         .LONG   $$MEDIA$$
              08  0004                         .BYTE   DT$_TU81
                  0005
                  00C0   948           MEDIA   <MU>, <TA78>
                  0005
        6D68104E  0005                         .LONG   $$MEDIA$$
              06  0009                         .BYTE   DT$_TA78
                  000A
                  00C0   949           MEDIA   <MU>, <TA81>
                  000A
        6D681051  000A                         .LONG   $$MEDIA$$
              09  000E                         .BYTE   DT$_TA81
                  000F
                  00C0   950           MEDIA   <MU>, <TK50>
                  000F
        6D68B032  000F                         .LONG   $$MEDIA$$
              0A  0013                         .BYTE   DT$_TK50
                  0014
                  00C0   951           MEDIA   <MF>, <TU78>
                  0014
        69A9504E  0014                         .LONG   $$MEDIA$$
              05  0018                         .BYTE   DT$_TU78
                  0019
```

```
                             00C0           .SBTTL  Controller Initialization Routine
                             00C0
                             00C0      ;+
                             00C0      ; MSCP speaking intelligent controller initialization routine.
                             00C0      ;
                             00C0      ; INPUTS:
                             00C0      ;        R4 => System ID of intelligent controller.
                             00C0      ;        R5 => IDB
                             00C0      ;        R6 => DDB
                             00C0      ;        R8 => CRB for intelligent controller.
                             00C0      ;
                             00C0
                             00C0      TU_CONTROLLER_INIT:
                06     11    00C0           BRB     0$                      ; Branch around breakpoint.
        00000000'GF    16    00C2           JSB     G^INI$BRK               ; Breakpoint for debugging.
                             00C8      0$:
                             00C8
                             00C8      ; Check for CDDB already present.  If a CDDB is present, this call results
                             00C8      ; from a power failure.  This driver performs power failure recovery as a
                             00C8      ; result of virtual circuit closure notification.  No action need be taken
                             00C8      ; here.
                             00C8
             10 A8     D5    00C8           TSTL    CRB$L_AUXSTRUC(R8)      ; Is there a CDDB present?
                01     13    00CB           BEQL    5$                      ; Branch if CDDB is not present.
                       05    00CD           RSB                             ; Else, just exit.
                             00CE
                             00CE      ; Check that only one UCB is chained onto the input DDB.  This UCB could be
                             00CE      ; the boot device UCB.  Therefore, make the UCB online so that I/O may be
                             00CE      ; performed on it.  All other initialization of the UCB is performed as the
                             00CE      ; result cf DPT_STORE entries place in the INIT section of the DPT by the
                             00CE      ; INIT_UCB macro.
                             00CE
                             00CE      5$:
          55   04 A6   D0    00CE           MOVL    DDB$L_UCB(R6),R5        ; R5 => first UCB if any.
          64 A5   10   C8    00D2           BISL    #UCB$M_ONLINE, -        ; Set the possibily boot UCB online.
                             00D6                   UCB$L_STS(R5)
             30 A5     D5    00D6           TSTL    UCB$L_LINK(R5)          ; Is there another UCB?
                04     13    00D9           BEQL    10$                     ; EQL implies no more UCB's.
                             00DB           BUG_CHECK       TAPECLASS,FATAL ; For now.
                             00DF      10$:
                             00DF
                             00DF      ; Setup those values which must be correct before IPL is lowered from 31.
                             00DF      ; Then FORK to create an IPL$_SCS fork thread which will complete controller
                             00DF      ; initialization.  Initialization of an MSCP server requires several message
                             00DF      ; exchanges and consumes several seconds.  Therefore, this work is conducted
                             00DF      ; in a fork thread with other system initialization proceeding concurrently.
                             00DF
          10 A8   55   D0    00DF           MOVL    R5, CRB$L_AUXSTRUC(R8)  ; The UCB will act as a CDDB until the
                             00E3                                           ; real one is built.
        00CC C5   64   7D    00E3           MOVQ    (R4), -                 ; Setup remote system ID for call to
                             00E8                   UCB$Q_UNIT_ID(R5)       ; DUTU$CREATE_CDDB.
                             00E8
                             00E8           FORK                            ; Create initialization fork thread.
                             00EE
                             00EE      ; Create and initialize the CDDB.
                             00EE
               FF0F'   30    00EE           BSBW    DUTU$CREATE_CDDB
```

```
                                  00F1      ;
                                  00F1      ; Here we call an internal subroutine which:
                                  00F1      ;
                                  00F1      ;         1. Makes a connection to the MSCP server in the intelligent
                                  00F1      ;                 controller.
                                  00F1      ;
                                  00F1      ;         2. Sends an MSCP command to SET CONTROLLER CHARACTERISTICS.
                                  00F1      ;
                                  00F1      ;         3. Allocates an MSCP buffer and RSPID for our future use in
                                  00F1      ;                 connection management.
                                  00F1      ;
                                  00F1      ; Upon return R4 => PDT and R5 => CDRP.
                                  00F1      ;
                                  00F1
               55    00D0 C5   DE 00F1              MOVAL   CDDB$A_PRMCDRP(R5), R5  ; Get permanent CDRP address.
                         0088  30 00F6              BSBW    MAKE_CONNECTION         ; Call internal subroutine to make
                                  00F9                                              ;  a connection to the MSCP server in
                                  00F9                                              ;  the intelligent controller. Input
                                  00F9                                              ;  and output are R5 => CDRP.
                                  00F9
                                  00F9              PERMCDRP_TO_CDDB -              ; Get CDDB address in R3.
                                  00F9                      cdrp=R5, cddb=R3
               50     18 A3    D0 0100              MOVL    CDDB$L_CRB(R3),R0       ; Get CRB address.
          1C A0   0EF0'CF      9E 0104              MOVAB   W^TU$TMR, -             ; Establish permanent timeout routine.
                                  010A                      CRB$L_TOUTROUT(R0)
               51     2A A3    3C 010A              MOVZWL  CDDB$W_CNTRLTMO(R3), R1 ; Get controller timeout interval.
  18 A0    00000000'GF    51   C1 010E              ADDL3   R1, G^EXE$GL_ABSTIM, -  ; Use that to set next timeout
                                  0117                      CRB$L_DUETIME(R0)       ; wakeup time.
                                  0117
                                  0117              ; The normal MSCP timeout mechanism is now in effect.  Henceforth,
                                  0117              ; no fork thread may use the CDDB permanent CDRP as a fork block.
                                  0117
                                  0117              ASSUME  CDDB$V_DAPBSY GE 8
            13 A3     04       88 0117              BISB    #<CDDB$M_DAPBSY @ -8>, -; Set DAP CDRP in use flag.
                                  011B                      CDDB$W_STATUS+1(R3)
               55     54 A3    D0 011B              MOVL    CDDB$L_DAPCDRP(R3), R5  ; Get DAP CDRP address.
                       FEDE'   30 011F              BSBW    DUTU$POLL_FOR_UNITS     ; Poll controller for units.
                                  0122
          12 A3    0080 8F     AA 0122              BICW    #CDDB$M_NOCONN, -       ; Now that connection is good, clear
                                  0128                      CDDB$W_STATUS(R3)       ; the no connection active bit.
                                  0128
     55    53   0000007C 8F    C3 0128              SUBL3   #<UCB$L_CDDB_LINK -     ; Get "previous" UCB address in R0.
                                  0130                       -CDDB$L_UCBCHAIN>, R3, R5
                                  0130
               55    00C4 C5   D0 0130      100$:   MOVL    UCB$L_CDDB_LINK(R5), R5 ; Link to next UCB (if any).
                          1A   13 0135              BEQL    120$                    ; EQL implies no more UCB's.
                                  0137              .IF     DEFINED TU_TRACE
                                  0137              BSBW    TRACE_INIT              ; Init IRP trace table.
                                  0137              .ENDC
          68 A5    0400 8F     AA 0137              BICW    #UCB$M_MSCP_WAITBMP, -  ; Indicate RWAITCNT no longer bumped.
                                  013D                      UCB$W_DEVSTS(R5)
                       56 A5   B7 013D              DECW    UCB$W_RWAITCNT(R5)      ; Decrement wait count to allow I/O.
                          03   13 0140              BEQL    110$                    ; Branch if wait count is zero.
                       FEBB'   30 0142              BSBW    DUTU$CHECK_RWAITCNT     ; Else, check wait count validity.
                          3F   BB 0145      110$:   PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save registers before call.
                 00000000'GF   16 0147              JSB     G^SCS$UNSTALLUCB        ; Startup any queued up I/O requests.
                          3F   BA 014D              POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore registers after call.
```

```
                DF   11 014F         BRB    100$                   ; Loop back to test more UCB's (if any).
   12 A3   0404 8F   AA 0151 120$:   BICW   #<CDDB$M_INITING -     ; Clear ''initing'' and DAP CDRP busy
                        0157                 !CDDB$M_DAPBSY>, -    ; flags.
                        0157                CDDB$W_STATUS(R3)
                     05 0157         RSB                           ; Terminate this thread of execution.
                        0158
                        0158 INIT_TIMEOUT:                         ; Controller Init Timeout handler.
              0BF0   31 0158         BRW    TU$RE_SYNCH            ; If we timeout, try to restart.
```

```
                                      015B          .SBTTL  MAKE_CONNECTION
                                      015B
                                      015B  ; MAKE_CONNECTION - Internal subroutine, called from TU_CONTROLLER_INIT and
                                      015B  ;       TU$CONNECT_ERR, that establishes a connection to the MSCP server
                                      015B  ;       in the intelligent controller.
                                      015B  ;
                                      015B  ; INPUTS:
                                      015B  ;       R5 => permanent CDRP
                                      015B  ;
                                      015B  ; OUTPUTS:
                                      015B  ;       Connection established and initial SET CONTROLLER CHARACTERISTICS
                                      015B  ;       command is sent to controller.  Also an MSCP buffer and an RSPID
                                      015B  ;       are allocated for the connection.
                                      015B  ;
                                      015B  ;       Side effects include the fact that all registers, except R5, are
                                      015B  ;       modified.
                                      015B  ;
                                      015B
5F 4C 43 5F 45 50 41 54 24 53 4D 56   015B  CLASS_DRVR_NAME:        .ASCII  /VMS$TAPE_CL_DRVR/
                        52 56 52 44   0167
20 20 20 45 50 41 54 24 50 43 53 4D   016B  MSCP_SRVR_NAME:         .ASCII  /MSCP$TAPE       /
                        20 20 20 20   0177
                                      017B
                                      017B  HSTIMEOUT_ARRAY:                        ; Host timeouts for various controllers.
                                      017B          ASSUME  MSCP$K_CM_HSC50 EQ      1
                                      017B          ASSUME  MSCP$K_CM_UDA50 EQ      2
                                      017B          ASSUME  MSCP$K_CM_RC25  EQ      3
                                      017B          ASSUME  MSCP$K_CM_EMULA EQ      4
                                      017B          ASSUME  MSCP$K_CM_TU81  EQ      5
                                      017B          ASSUME  MSCP$K_CM_UDA52 EQ      6
                                 1E   017B          .BYTE   HOST_TIMEOUT            ; Use default constant for HSC50.
                                 00   017C          .BYTE   0                       ; Use zero for dedicated controller.(UDA50)
                                 00   017D          .BYTE   0                       ; Use zero for dedicated controller.(AZTEC)
                                 1E   017E          .BYTE   HOST_TIMEOUT            ; Use default constant for Emulator.
                                 00   017F          .BYTE   0                       ; Use zero for dedicated controller.(TU81)
                                 00   0180          .BYTE   0                       ; Use zero for dedicated controller.(UDA52)
                                      0181
                                      0181  MAKE_CONNECTION:
                                      0181
                                      0181          PERMCDRP_TO_CDDB -              ; Get CDDB address from CDRP.
                                      0181                  cdrp=R5, cddb=R2
                         44 A2 8ED0   0188          POPL    CDDB$L_SAVED_PC(R2)     ; Save caller's return in CDDB field.
                                      018C  5$:
                    00000000'GF  D0   018C          MOVL    G^EXE$GL_ABSTIM,-       ; Copy absolute time that we entered
                          30 A2       0192                  CDDB$L_OLDCMDSTS(R2)    ;  this routine, or the last time that
                                      0194                                          ;  terminated all pending I/O.
                                      0194  10$:
             50     00000000'GF  D0   0194          MOVL    G^SGN$GL_VMSD3,R0       ; Pickup interval of seconds that we
                                      019B                                          ;  should try to CONNECT until we
                                      019B                                          ;  decide to terminate pending I/O.
                             12  13   019B          BEQL    15$                     ; EQL implies infinite timeout.
                       50 30 A2  C0   019D          ADDL    CDDB$L_OLDCMDSTS(R2),R0 ; Sum is end of timeout interval.
                    00000000'GF 50  D1   01A1       CMPL    R0,G^EXE$GL_ABSTIM      ; See if we have timed out.
                             05  14   01A8          BGTR    15$                     ; GTR means no, time remains.
                           0150  30   01AA          BSBW    TERMINATE_PENDING       ; Else call to terminate all pending I/O
                             DD  11   01AD          BRB     5$                      ; Loop back to establish a new timeout
                                      01AF                                          ;  period.
```

```
                                01AF  15$:
                                01AF          CONNECT TU$IDR,-                  ; Entry point of Input Dispatcher Routine.
                                01AF                  TU$DGDR,-                 ; Entry point of Datagram Dispatcher.
                                01AF                  TU$CONNECT_ERR,-          ; Error entry point.
                                01AF                  CDDB$B_SYSTEMID(R2),-     ; Destination SYSTEM ID.
                                01AF                  ,-                        ; Remote station address.
                                01AF                  MSCP_SRVR_NAME,-          ; MSCP server name.
                                01AF                  CLASS_DRVR_NAME,-         ; Ascii of class driver name.
                                01AF                  #INITIAL_CREDIT,-         ;  Needs definition
                                01AF                  #MIN_SEND_CREDIT,-        ; Minimum send credit
                                01AF                  #INITIAL_DG_COUNT,-       ; Initial DataGram count
                                01AF                  ,-                        ; Block transfer priority
                                01AF                  ,-                        ; Connect data
                                01AF                  (R2),-                    ; Also pass CDDB address to CDT$L_AUXSTRUC
                                01AF                  ,                         ; Bad Response packet address
                                01E7
            28 50        E8     01E7          BLBS    R0,30$                    ; LBS implies success, so branch around.
                                01EA
      52    08 A5        32     01EA          CVTWL   CDRP$W_CDRPSIZE(R5),R2    ; R2 has negative offset, from base of
                                01EE                                            ;  CDRP, of base of CDDB.
         52    55        C0     01EE          ADDL    R5,R2                     ; R2 => CDDB.
      53    18 A2        D0     01F1          MOVL    CDDB$L_CRB(R2),R3         ; R3 => CRB.
                                01F5
   1C A3    04'AF        9E     01F5          MOVAB   B^20$,CRB$L_TOUTROUT(R3); Establish LABEL as place to call, for
                                01FA                                            ;  now, for periodic wakeups.
                  0A     C1     01FA          ADDL3   #CONNECT_DELTA,-          ; Establish Due time as a little in
       00000000'GF              01FC                  G^EXE$GL_ABSTIM,-         ;  the future.
             18 A3              0201                  CRB$L_DUETIME(R3)
                         05     0203          RSB                               ; Return to caller's caller and kill
                                0204                                            ;  this thread.
                                0204  20$:
      52    10 A3        D0     0204          MOVL    CRB$L_AUXSTRUC(R3),R2     ; R2 => CDDB.
    55    00D0 C2        9E     0208          MOVAB   CDDB$A_PRMCDRP(R2),R5     ; Get permanent CDRP address.
                                020D          SETIPL  #IPL$_SCS                 ; Lower IPL after wakeup.
                  82     11     0210          BRB     10$                       ; Loop back and try CONNECT again.
                                0212  30$:    ; A connection has been established
                                0212          PERMCDRP_TO_CDDB -                ; Get CDDB address from CDRP.
                                0212                  cdrp=R5, cddb=R1
   00F4 C1       53      D0     0219          MOVL    R3, CDDB$L_CDT(R1)        ; Save CDT address (in perm CDRP).
     14 A1       54      D0     021E          MOVL    R4, CDDB$L_PDT(R1)        ; Save PDT address.
   01B8 C1       53      D0     0222          MOVL    R3, CDDB$L_DAPCDT(R1)     ; Save CDT address in DAP CDRP too.
        53       51      D0     0227          MOVL    R1, R3                    ; Now that CDT is saved, move CDDB addr.
                                022A
      51    18 A3        D0     022A          MOVL    CDDB$L_CRB(R3), R1        ; Get CRB address.
      18 A1      01      CE     022E          MNEGL   #1, CRB$L_DUETIME(R1)     ; Infinite time till next timeout, now.
   1C A1    FF22 CF      9E     0232          MOVAB   INIT_TIMEOUT, -           ; Establish timeout routine that will
                                0238                  CRB$L_TOUTROUT(R1)        ; serve for rest of controller init.
                                0238
                                0238  ;
                                0238  ; Here we prepare to send a SET CONTROLLER CHARACTERISTICS MSCP Packet to
                                0238  ;       the intelligent controller over the connection that we have just
                                0238  ;       established.
                                0238  ;
                                0238
                                0238          ALLOC_RSPID                       ; ALLOCate a ReSPonse ID.
                                023E          ALLOC_MSG_BUF                     ; Allocate an MSCP buffer (and also
                                0241                                            ;  allocate a unit of flow control).
        07 50   E8  0241               BLBS    R0,50$                  ; If success, branch around.
   53   18 A3   D0  0244               MOVL    CDDB$L_CRB(R3),R3       ; TU$RE_SYNCH expects R3 => CDDB.
         0B00   31  0248               BRW     TU$RE_SYNCH             ; Failure here means we must re-CONNECT.
                    024B       50$:                                    ; Here R2 => MSCP buffer allocated.
           51   D4  024B               CLRL    R1                      ; First set Controller Characteristics
                    024D                                               ;  with zero (i.e. infinite) host timeout.
           3C   10  024D               BSBB    PRP_STCON_MSG           ; Call to prepare MSCP command.
                    024F               SEND_MSCP_MSG_DRIVER            ; Returns with end-message addr. in R2.
         006A   30  0252               BSBW    RECORD_STCON            ; Record Controller Characteristics.
                    0255
                    0255               RECYCH_MSG_BUF                  ; We recycle the END PACKET and
                    0258                                               ;  thereby allocate a new send credit.
                    0258               RECYCL_RSPID                    ; We also recycle the RSPID.
                    025E
                    025E       ; Determine the correct host timeout interval.  This is the larger of
                    025E       ; HSTIMEOUT_ARRAY[controller_model] and the controller timeout interval
                    025E       ; returned by the just completed Set Controller Characteristics.  There is,
                    025E       ; however, one wrinkle.  Zero represents an infinite timeout and therefore is
                    025E       ; larger than any other number.  Also, the controller already believes the
                    025E       ; host timeout interval to be infinite, as the result of the previous Set
                    025E       ; Controller Characteristics command.  Therefore, no further action need be
                    025E       ; taken when the timeout interval is infinite.
                    025E
   51   26 A3   9A  025E               MOVZBL  CDDB$B_CNTRLMDL(R3),R1  ; Get controller model type.
51   FF13 CF41  9A  0262               MOVZBL  HSTIMEOUT_ARRAY-1[R1],R1; Get corresponding host timeout value.
           1E   13  0268               BEQL    60$                     ; If zero, branch around.
   50   2A A3   3C  026A               MOVZWL  CDDB$W_CNTRLTMO(R3), R0 ; Get controller timeout interval.
           18   13  026E               BEQL    60$                     ; If controller timeout is infinite,
                    0270                                               ; use already set infinite host timeout.
      51   50   D1  0270               CMPL    R0, R1                  ; Compare with HSTIMEOUT_ARRAY value.
           03   1F  0273               BLSSU   55$                     ; Branch if HSTIMEOUT_ARRAY is larger.
      51   50   D0  0275               MOVL    R0, R1                  ; Else, use controller timeout as
                    0278                                               ; host timeout interval.
                    0278       55$:
           11   10  0278               BSBB    PRP_STCON_MSG           ; Else reset controller characteristics.
                    027A               SEND_MSCP_MSG_DRIVER            ; Returns with end-message addr. in R2.
           40   10  027D               BSBB    RECORD_STCON            ; Record Controller Characteristics.
                    027F               RECYCH_MSG_BUF                  ; Again we recycle the END PACKET and
                    0282                                               ;  thereby allocate a new send credit.
                    0282               RECYCL_RSPID                    ; We also recycle the RSPID.
                    0288       60$:
                    0288
        44 B3   17  0288               JMP     @CDDB$L_SAVED_PC(R3)    ; Return to caller.
```

```
                        028B    ; PRP_STCON_MSG - Prepare a Set Controller Characteristics Command Message.
                        028B    ;
                        028B    ; Inputs:
                        028B    ;        R1 =  Host Timeout Value
                        028B    ;        R2 => MSCP buffer to fill
                        028B    ;        R3 => CDDB
                        028B    ;        R5 => CDRP
                        028B    ;
                        028B
                        028B   PRP_STCON_MSG:
                        028B
           51     DD    028B           PUSHL   R1                         ; Save important register.
                        028D           INIT_MSCP_MSG                      ; Initialize buffer for MSCP message.
           51   8ED0    0290           POPL    R1                         ; Restore important register.
                        0293
           04     90    0293           MOVB    #MSCP$K_OP_STCON,-         ; Insert SET CONTROLLER CHARACTERISTICS
        08 A2           0295                   MSCP$B_OPCODE(R2)          ;  opcode with NO modifiers.
                        0297
        28 A3     B0    0297           MOVW    CDDB$W_CNTRLFLGS(R3),-     ; Set host settable characteristics
        0E A2           029A                   MSCP$W_CNT_FLGS(R2)        ;  bits into MSCP command message.
                        029C
  10 A2    51     B0    029C           MOVW    R1,MSCP$W_HST_TMO(R2)      ; Set host timeout into MSCP packet.
                        02A0
 00000000'GF      7D    02A0           MOVQ    G^EXE$GQ_SYSTIME,-         ; Transmit time of century in clunks.
        14 A2           02A6                   MSCP$Q_TIME(R2)
                        02A8
 50     18 A3     D0    02A8           MOVL    CDDB$L_CRB(R3),R0          ; R0 => CRB.
 7E     2A A3     3C    02AC           MOVZWL  CDDB$W_CNTRLTMO(R3),-(SP); Pickup controller delta.
           03     12    02B0           BNEQ    70$                        ; NEQ implies this controller has been
                        02B2                                              ;   init'ed at least once before.
     6E    1E     D0    02B2           MOVL    #INIT_IMMED_DELTA,(SP)     ; Else use compiled in timeout.
                        02B5   70$:
           8E     C1    02B5           ADDL3   (SP)+,-                    ; Establish delta time for time out
 00000000'GF            02B7                   G^EXE$GL_ABSTIM,-          ;  to prevent against controller never
        18 A0           02BC                   CRB$L_DUETIME(R0)          ;  responding.
                        02BE
                  05    02BE           RSB                                ; Return to caller.
```

```
                              02BF  ; RECORD_STCON - Record data from a Set Controller Characteristics end message
                              02BF  ;                in the CDDB.
                              02BF  ;
                              02BF  ; Inputs:
                              02BF  ;        R2 => MSCP End Message
                              02BF  ;        R3 => CDDB
                              02BF
                              02BF
                              02BF RECORD_STCON:
               0E A2    B0    02BF         MOVW    MSCP$W_CNT_FLGS(R2),-    ; Pickup NON-host settable characteristics
               28 A3          02C2                 CDDB$W_CNTRLFLGS(R3)     ;  from END PACKET and save in CDDB.
                              02C4
               10 A2    B0    02C4         MOVW    MSCP$W_CNT_TMO(R2),-     ; Likewise with controller timeout.
               2A A3          02C7                 CDDB$W_CNTRLTMO(R3)
                              02C9
               14 A2    7D    02C9         MOVQ    MSCP$Q_CNT_ID(R2),-      ; Also save controller unique ID.
               20 A3          02CC                 CDDB$Q_CNTRLID(R3)
                              02CE
      29 12 A3    06    E2    02CE         BBSS    #CDDB$V_ALCLS_SET, -     ; Branch if allocation class already
                              02D3                 CDDB$W_STATUS(R3), 90$   ; set, and indicate it is now set.
                              02D3         ; The allocation class is about to be set for this device.  The object
                              02D3         ; is to give every reasonable chance for the value to be non-zero.
 50 A3    00000000'GF   D0    02D3         MOVL    G^CLU$GL_ALLOCLS, -      ; Assume a local, single host
                              02DB                 CDDB$L_ALLOCLS(R3)       ; controller.
         26 A3    01    91    02DB         CMPB    #MSCP$K_CM_HSC50, -      ; Is this an HSC?
                              02DF                 CDDB$B_CNTRLMDL(R3)
                  05    13    02DF         BEQL    1099$                    ; Branch to multihost leg, if HSC.
      05 28 A3    02    E1    02E1         BBC     #MSCP$V_CF_MLTHS, -      ; Branch if a single host controller.
                              02E6                 CDDB$W_CNTRLFLGS(R3), -
                              02E6                 80$
                              02E6 1099$:
      50 A3    04 A2    9A    02E6         MOVZBL  MSCP$B_CNT_ALCS(R2), -   ; Get set controller characteristics
                              02EB                 CDDB$L_ALLOCLS(R3)       ; allocation class.
         50    E4 A3    9E    02EB 80$:    MOVAB   <CDDB$L_DDB -            ; Init loop through all DDBs.
                              02EF                 -DDB$L_CONLINK>(R3), R0
         50    38 A0    D0    02EF 82$:    MOVL    DDB$L_CONLINK(R0), R0    ; Link to next DDB.
                  07    13    02F3         BEQL    90$                      ; Branch if no more DDBs.
      3C A0    50 A3    D0    02F5         MOVL    CDDB$L_ALLOCLS(R3), -    ; Copy allocation class to this
                              02FA                 DDB$L_ALLOCLS(R0)        ; DDB.
                  F3    11    02FA         BRB     82$                      ; Loop till no more DDBs.
                              02FC
                        05    02FC 90$:    RSB
```

```
                              02FD  1314          .SBTTL  TERMINATE_PENDING
                              02FD  1315
                              02FD  1316  ; TERMINATE_PENDING - internal routine called from MAKE_CONNECTION.
                              02FD  1317  ;       The purpose of this routine is to terminate all pending I/O on
                              02FD  1318  ;       this connection because the amount of time specified in a SYSGEN
                              02FD  1319  ;       parameter has passed without being able to CONNECT.
                              02FD  1320  ;
                              02FD  1321  ; Inputs:
                              02FD  1322  ;       R2 => CDDB
                              02FD  1323  ;       R5 => CDRP
                              02FD  1324  ;
                              02FD  1325  ; Outputs:
                              02FD  1326  ;       Registers R0, R1, R3 are modified.
                              02FD  1327  ;
                              02FD  1328
                              02FD  1329 TERMINATE_PENDING:
                  02     E0   02FD  1330         BBS     #CDDB$V_INITING,-            ; Do not time out during initialization.
            3D 12 A2          02FF  1331                 CDDB$W_STATUS(R2),50$
                              0302  1332 10$:
         50    3C B2     0F   0302  1333         REMQUE  @CDDB$L_RSTRTQFL(R2),R0 ; REMQUE a pending CDRP. R0 => CDRP.
                  0F     1D   0306  1334         BVS     20$                          ; VS implies queue empty.
                              0308  1335         POST_CDRP status=SS$_CTRLERR         ; Terminate this CDRP.
                  EB     11   0315  1336         BRB     10$                          ; Loop thru all CDRP's on CDDB Q.
                              0317  1337 20$:
   52    0000007C 8F     C3   0317  1338         SUBL3   #<UCB$L_CDDB_LINK -          ; Get "previous" UCB in R3.
                              031E  1339                  -CDDB$L_UCBCHAIN>, -
                  53          031E  1340                 R2, R3
                              031F  1341
       53    00C4 C3     D0   031F  1342 30$:    MOVL    UCB$L_CDDB_LINK(R3), R3 ; Chain to next UCB (if any).
                  19     13   0324  1343         BEQL    50$                          ; EQL implies no more UCB's here.
                              0326  1344 40$:
         50    4C B3     0F   0326  1345         REMQUE  @UCB$L_IOQFL(R3),R0          ; R0 => IRP on Q.
                  F3     1D   032A  1346         BVS     30$                          ; VS implies I/O queue empty.
         50    60 A0     9E   032C  1347         MOVAB   -CDRP$L_IOQFL(R0),R0         ; R0 => CDRP portion of IRP.
                              0330  1348         POST_CDRP status=SS$_CTRLERR         ; Terminate this CDRP.
                  E7     11   033D  1349         BRB     40$                          ; Loop thru all IRP's on UCB.
                              033F  1350 50$:
                         05   033F  1351         RSB                                  ; Return to caller.
```

```
                          0340                .SBTTL  BRING_UNIT_ONLINE
                          0340
                          0340   ; BRING_UNIT_ONLINE - Internal subroutine to bring an available unit online.
                          0340   ;           This subroutine is called from TU$CONNECT_ERR.
                          0340   ;
                          0340   ; INPUTS:
                          0340   ;           R3 => CDDB
                          0340   ;           R4 => PDT
                          0340   ;           R5 => UCB
                          0340   ;
                          0340   ; Implicit Inputs:
                          0340   ;
                          0340   ;           CDDB$W_STATUS(R3)          CDDB$V_DAPBSY set
                          0340   ;
                          0340   ;           The normal class driver MSCP operation timeout mechanism must be
                          0340   ;           enabled.
                          0340   ;
                          0340
                          0340   BRING_UNIT_ONLINE:
                          0340
           44 A3 8ED0     0340          POPL    CDDB$L_SAVED_PC(R3)     ; Save caller's return address.
   50      54 A3   D0     0344          MOVL    CDDB$L_DAPCDRP(R3), R0  ; Get DAP CDRP address.
        53    55   D0     0348          MOVL    R5, R3                  ; Copy UCB address.
        55    50   D0     034B          MOVL    R0, R5                  ; Copy CDRP address.
                          034E
   BC A5      53   D0     034E          MOVL    R3, CDRP$L_UCB(R5)      ; Setup UCB address in CDRP.
                          0352
                          0352          ALLOC_MSG_BUF                   ; Allocate a message buffer.
           01 50   E8     0355          BLBS    R0, 3$                  ; Branch if connection is not broken.
                   05     0358          RSB                             ; Else, just kill this fork thread.
                          0359   3$:    ALLOC_RSPID                     ; Allocate a response-id.
                          035F          INIT_MSCP_MSG ucb=(R3)          ; Initialize buffer for MSCP message.
                          0362
              09   90     0362          MOVB    #MSCP$K_OP_ONLIN,-      ; ONLINE command, zero modifiers.
           08 A2          0364                  MSCP$B_OPCODE(R2)
                          0366
                   A8     0366          BISW    #MSCP$M_MD_CLSEX-       ; Do exclusive ONLINE and clear serious
                          0367                  !MSCP$M_MD_EXCLU,-      ;  exception.
0A A2    2020 8F          0367                  MSCP$W_MODIFIER(R2)
                          036C
         00E0 C3   B0     036C          MOVW    UCB$W_UNIT_FLAGS(R3),-  ; Copy UNIT flags to MSCP packet.
           0E A2          0370                  MSCP$W_UNT_FLGS(R2)
                          0372
         00D8 C3   D0     0372          MOVL    UCB$L_MSCPDEVPARAM(R3),-; Copy Device dependent parameters to
           1C A2          0376                  MSCP$L_DEV_PARM(R2)     ;  MSCP packet.
                          0378
              08   EF     0378          EXTZV   #MT$V_DENSITY,-         ; Determine density that the user has
              05          037A                  #MT$S_DENSITY,-         ;  last established for this unit
   50      44 A3          037B                  UCB$L_DEVDEPEND(R3),R0  ;  and put into R0.
                          037E
            008B   30     037E          BSBW    VMSTOMSCP_DENS          ; Convert VMS density to MSCP format.
   20 A2      51   B0     0381          MOVW    R1,MSCP$W_FORMAT(R2)    ; Move MSCP density in R1 into packet.
                          0385
              05   E1     0385          BBC     #MSCP$V_UF_VSMSU,-      ; Test if we are suppressing variable
     0D 0E A2             0387                  MSCP$W_UNT_FLGS(R2),10$ ;  speed mode, and branch if NOT.
              18   EF     038A          EXTZV   #MT$V_SPEED,-           ; Extract user's speed specification
              08          038C                  #MT$S_SPEED,-           ;  from UCB.
```

```
 50   44 A3          038D  1410              UCB$L_DEVDEPEND(R3),R0  ;  and put into R0.
        009D    30   0390  1411        BSBW    SPEEDTOMSCP
 22 A2    50    B0   0393  1412        MOVW    R0,MSCP$W_SPEED(R2)     ; Move MSCP speed in R0 into packet.
                     0397  1413
                     0397  1414 10$:   SEND_MSCP_MSG_DRIVER            ; ONLIN - returns end pkt. addr. in R2.
                     039A  1415        IF_MSCP FAILURE, then=30$       ; Branch if ONLIN failed.
                     03A0  1416
                     03A0  1417 ; If here then various fields in the END PACKET are valid.
                     03A0  1418 ;      Here we have just brought ONLINE a unit that was online before
                     03A0  1419 ;      as a result of a failed previous CONNECTION.  We assume
                     03A0  1420 ;      that the volume is identical to the one that was ONLINE here before.
                     03A0  1421 ;      And then setup the UCB accordingly.
                     03A0  1422 ;
                     03A0  1423
         03F2   30   03A0  1424        BSBW    RECORD_ONLINE           ; Move data from end message to UCB.
                     03A3  1425
                     03A3  1426        RESET_MSCP_MSG                  ; Setup message buf. etc. for reuse.
                     03A6  1427
           03   90   03A6  1428        MOVB    #MSCP$K_OP_GTUNT,-      ; GET UNIT STATUS command, zero modifiers.
        08 A2        03A8  1429                MSCP$B_OPCODE(R2)
                     03AA  1430
                     03AA  1431        SEND_MSCP_MSG_DRIVER            ; GTUNT - returns end pkt. addr. in R2.
                     03AD  1432        IF_MSCP FAILURE, then=30$       ; Branch if GTUNT failed.
                     03B3  1433
         03ED   30   03B3  1434        BSBW    RECORD_GETUNIT_CHAR     ; Record UNIT status data in UCB.
                     03B6  1435
                     03B6  1436 ; Here reposition out to where we were before.
                     03B6  1437
                     03B6  1438        RESET_MSCP_MSG                  ; Setup message buf. etc. for reuse.
                     03B9  1439
           25   90   03B9  1440        MOVB    #MSCP$K_OP_REPOS,-      ; Reposition command.
        08 A2        03BB  1441                MSCP$B_OPCODE(R2)
                A8   03BD  1442        BISW    #MSCP$M_MD_REWND-       ; Rewind and then space out an absolute
                     03BE  1443                !MSCP$M_MD_OBJCT,-      ;  number of objects.
 0A A2    06         03BE  1444                MSCP$W_MODIFIER(R2)
   00B0 C3      D0   03C1  1445        MOVL    UCB$L_RECORD(R3),-      ; Copy number of objects (gaps) to skip
        0C A2        03C5  1446                MSCP$L_REC_CNT(R2)      ;  into MSCP command packet.
                     03C7  1447
                     03C7  1448        SEND_MSCP_MSG_DRIVER            ; REPOS - returns end pkt. addr. in R2.
                     03CA  1449        IF_MSCP FAILURE, then=30$       ; Branch if REPOS failed.
                     03D0  1450
        FC2D'   30   03D0  1451 20$:   BSBW    DUTU$DEALLOC_ALL        ; Deallocate all CDRP resources.
                     03D3  1452
                     03D3  1453        PERMCDRP_TO_CDDB -              ; Get CDDB address in R3.
                     03D3  1454                cdrp=R5, cddb=R3
 55   BC A5     D0   03DA  1455        MOVL    CDRP$L_UCB(R5), R5      ; Restore input UCB address.
      44 B3     17   03DE  1456        JMP     @CDDB$L_SAVED_PC(R3)    ; Return to caller.
                     03E1  1457
                     03E1  1458 30$:                                   ; HERE if volume has changed.
                     03E1  1459        ASSUME  UCB$V_VALID GE 8
 65 A3    08    8A   03E1  1460        BICB    #<UCB$M_VALID @ -8>, -  ; If could not put the drive ONLINE,
                     03E5  1461                UCB$W_STS+1(R3)         ; clear the volume valid bit.
           07   E1   03E5  1462        BBC     #MSCP$V_SC_DUPUN,-      ; Branch around if NOT duplicate
     03 0A A2        03E7  1463                MSCP$W_STATUS(R2),40$   ;  unit substatus.
        FC13'   30   03EA  1464        BSBW    DUTU$SEND_DUPLICATE_UNIT; Notify operator of duplicate unit.
                     03ED  1465 40$:
                     03ED  1466        RESET_MSCP_MSG                  ; Setup message buf. etc. for reuse.
```

```
                          08   90  03F0  1467          MOVB    #MSCP$K_OP_AVAIL,-          ; Available command
                       08 A2       03F2  1468                  MSCP$B_OPCODE(R2)
                                   03F4  1469          SEND_MSCP_MSG DRIVER                ; AVAIL - returns end pkt. addr. in R2.
                          D7   11  03F7  1470          BRB     20$                         ; Join common exit code.
```

```
03F9                   .IF     DF      TU_SEQCHK
03F9                   .SBTTL  -       OVERRIDE_SEQCHK and REMOVE_SEQARY
03F9
03F9         ;+
03F9         ; OVERRIDE_SEQCHK - Set UCB$M_TU_OVRSQCHK bit in UCB$W_DEVSTS and then fall
03F9         ;                   thru to
03F9         ; REMOVE_SEQARY - Remove this IRP$L_SEQNUM from the UCB$L_TU_SEQARY and
03F9         ;                   collapse the array.
03F9         ;
03F9         ; Inputs:
03F9         ;       R5 => CDRP
03F9         ;
03F9
03F9         OVERRIDE_SEQCHK:
03F9
03F9                   PUSHL   R0                      ; Save R0.
03F9                   MOVL    CDRP$L_UCB(R5),R0       ; R0 => UCB.
03F9                   BISW    #UCB$M_TU_OVRSQCHK,-    ; Set bit to override sequence
03F9                           UCB$W_DEVSTS(R0)        ;  checking on this operation.
03F9                   POPL    R0                      ; Restore R0.
03F9
03F9         REMOVE_SEQARY:
03F9
03F9                   MOVQ    R0,-(SP)                ; Save registers.
03F9                   PUSHL   R3                      ;
03F9                   MOVL    CDRP$L_UCB(R5),R3       ; R3 => UCB.
03F9                   EXTZV   #0,#6,-                 ; Extract index of oldest array slot.
03F9                           UCB$B_TU_OLDINX(R3),R0
03F9                   EXTZV   #0,#6,-                 ; Extract index of next  array slot.
03F9                           UCB$B_TU_NEWINX(R3),R1
03F9         10$:
03F9                   EXTZV   #0,#6,R0,R0             ; Reduce R0 to 6-bit index.
03F9                   CMPL    R0,R1                   ; Have we run thru entire array?
03F9                   BEQL    50$                     ; EQL implies yes.
03F9                   CMPL    CDRP$L_SEQNUM(R5),-     ; If not, is this array slot ours?
03F9                           UCB$L_TU_SEQARY(R3)[R0]
03F9                   BEQL    20$                     ; EQL implies YES.
03F9                   INCL    R0                      ; Bump index.
03F9                   BRB     10$                     ; And continue loop.
03F9         20$:                                      ; Here R0 has array slot index.
03F9                   EXTZV   #0,#6,-                 ; Extract index of oldest array slot.
03F9                           UCB$B_TU_OLDINX(R3),-(SP)
03F9         30$:                                      ; Here we collapse the array by moving
03F9                                                   ;  each slot preceeding the slot to
03F9                                                   ;  remove, one position forward.  We
03F9                                                   ;  begin with the slot immediately
03F9                                                   ;  preceeding the found one.
03F9                   EXTZV   #0,#6,R0,R0             ; Reduce R0 to 6-bit index.
03F9                   CMPL    R0,(SP)                 ; Are we done?
03F9                   BEQL    40$                     ; EQL implies we are done.
03F9                   SUBL3   #1,R0,R1                ; R1 has index of preceeding slot.
03F9                   EXTZV   #0,#6,R1,R1             ; Reduce R1 to 6-bit index.
03F9                   MOVL    UCB$L_TU_SEQARY(R3)[R1],-       ; Move slot contents forward one
03F9                           UCB$L_TU_SEQARY(R3)[R0]         ;  position.
03F9                   DECL    R0                      ; Decrement index.
03F9                   BRB     30$                     ; And continue in loop.
03F9         40$:
```

```
                03F9            INCB    UCB$B_TU_OLDINX(R3)     ; Increment index to reflect collapse.
                03F9            TSTL    (SP)+                   ; Remove junk from stack.
                03F9       50$:
                03F9            POPL    R3                      ; Restore registers.
                03F9            MOVQ    (SP)+,R0
                03F9            RSB                             ; Return to caller.
                03F9            .ENDC
```

```
                                03F9  1538          .SBTTL  Density and Speed Conversion Routines
                                03F9  1539
                                03F9  1540  ;+
                                03F9  1541  ; VMSTOMSCP_DENS - Internal subroutine to convert from a VMS density
                                03F9  1542  ;       code to a MSCP density code.
                                03F9  1543  ;
                                03F9  1544  ; Inputs:
                                03F9  1545  ;       R0 = VMS density code
                                03F9  1546  ;
                                03F9  1547  ; Outputs:
                                03F9  1548  ;       R1 = MSCP density code
                                03F9  1549  ;               R0 = 0 which implies that the VMS code was such that we chose
                                03F9  1550  ;                       the default MSCP code
                                03F9  1551  ;               R0 = 1 which implies that the VMS code was a perfect match for
                                03F9  1552  ;                       one of the codes.
                                03F9  1553
                                03F9  1554  TU_VMSDENS:
                            03  03F9  1555                  .BYTE   MT$K_NRZI_800
                            04  03FA  1556                  .BYTE   MT$K_PE_1600
                            05  03FB  1557                  .BYTE   MT$K_GCR_6250
                            04  03FC  1558                  .BYTE   MT$K_PE_1600    ; Redundant for NOT FOUND case default.
                                03FD  1559
                                03FD  1560  TU_MSCPDENS:
                            01  03FD  1561                  .BYTE   MSCP$M_TF_800
                            02  03FE  1562                  .BYTE   MSCP$M_TF_PE
                            04  03FF  1563                  .BYTE   MSCP$M_TF_GCR
                                0400  1564
                                0400  1565  TU_ABSDENS:
                          0320  0400  1566                  .WORD   800
                          0640  0402  1567                  .WORD   1600
                          186A  0404  1568                  .WORD   6250
                          0640  0406  1569                  .WORD   1600            ; Redundant for NOT FOUND case.
                                0408  1570
                                0408  1571  TU_ABSPEED:
                            19  0408  1572                  .BYTE   25
                            4B  0409  1573                  .BYTE   75
                            7D  040A  1574                  .BYTE   125
                            FF  040B  1575                  .BYTE   255
                                040C  1576
                                040C  1577  VMSTOMSCP_DENS:
                                040C  1578
                                040C  1579          ASSUME  MT$K_NRZI_800   EQ      3
                                040C  1580          ASSUME  MT$K_PE_1600    EQ      4
                                040C  1581          ASSUME  MT$K_GCR_6250   EQ      5
                                040C  1582
         51     50     03   C3  040C  1583          SUBL3   #3,R0,R1                ; Subtract out NRZI bias from VMS code.
                       08   19  0410  1584          BLSS    10$                     ; LSS implies input NOT valid VMS code.
                50     01   D0  0412  1585          MOVL    #1,R0                   ; Setup for possible success return.
                03     51   D1  0415  1586          CMPL    R1,#3                   ; See if input in range.
                       05   19  0418  1587          BLSS    20$                     ; LSS implies yes.
                                041A  1588  10$:
                       50   D4  041A  1589          CLRL    R0                      ; Indicate we picked up default.
                51     01   D0  041C  1590          MOVL    #1,R1                   ; Default is MSCP 1600 bpi.
                                041F  1591  20$:
         51     DA AF41     9B  041F  1592          MOVZBW  TU_MSCPDENS[R1],R1      ; Extract MSCP code from array.
                            05  0424  1593          RSB                             ; Return to caller.
                                0425  1594
                          0425  ;+
                          0425  ; MSCPTOVMS_DENS - Internal routine to convert from MSCP density code to
                          0425  ;       VMS density code.
                          0425  ;
                          0425  ; Inputs:
                          0425  ;       R0 = MSCP density code
                          0425  ;
                          0425  ; Outputs:
                          0425  ;       R0 = VMS density code
                          0425  ;-
                          0425
                          0425  MSCPTOVMS_DENS:
                          0425
                          0425          ASSUME  MSCP$V_TF_800   EQ      0
                          0425          ASSUME  MSCP$V_TF_PE    EQ      1
                          0425          ASSUME  MSCP$V_TF_GCR   EQ      2
 50   50   03   00   EA   0425          FFS     #0,#3,R0,R0                     ; R0 contains 0, 1 or 2 (or 3 if not
                          042A                                                  ;  found).
      50   CB AF40   9A   042A          MOVZBL  TU_VMSDENS[R0],R0               ; R0 contains system density code.
                     05   042F          RSB                                     ; Return to caller.
                          0430
                          0430  ;+
                          0430  ; SPPEDTOMSCP - internal routine to calculate MSCP speed value.
                          0430  ;
                          0430  ; Inputs:
                          0430  ;       R0 = Speed in IPS
                          0430  ;       R1 = MSCP density value
                          0430  ;
                          0430  ; OUTPUTS:
                          0430  ;       R0 = MSCP speed value
                          0430  ;       R1 modified
                          0430  ;-
                          0430
                          0430  SPEEDTOMSCP:
                          0430
                          0430          ASSUME  MSCP$V_TF_800   EQ      0
                          0430          ASSUME  MSCP$V_TF_PE    EQ      1
                          0430          ASSUME  MSCP$V_TF_GCR   EQ      2
 51   51   03   00   EA   0430          FFS     #0,#3,R1,R1                     ; R1 contains 0, 1 or 2 (or 3 if not
                          0435                                                  ;  found).
      51   C7 AF41   3C   0435          MOVZWL  TU_ABSDENS[R1],R1               ; R1 contains system density code.
           50   51   C4   043A          MULL    R1,R0                           ; R0 contains absolute data rate.
 50   000003E8 8F    C6   043D          DIVL    #1000,R0                        ; MSCP value is rate/1000.
                     05   0444          RSB                                     ; Return to caller.
                          0445
                          0445  ;+
                          0445  ; MSCPTOSPEED - internal routine to convert MSCP data rate to speed in IPS.
                          0445  ;
                          0445  ; Inputs:
                          0445  ;       R0 = MSCP Data Rate
                          0445  ;       R1 = MSCP density value
                          0445  ;
                          0445  ; OUTPUTS:
                          0445  ;       R0 = MSCP speed value
                          0445  ;       R1 modified
                          0445  ;-
                          0445
```

```
                               0445  MSCPTOSPEED:
                               0445
                               0445             ASSUME  MSCP$V_TF_800   EQ      0
                               0445             ASSUME  MSCP$V_TF_PE    EQ      1
                               0445             ASSUME  MSCP$V_TF_GCR   EQ      2
     51   51   03   00    EA   0445             FFS     #0,#3,R1,R1             ; R1 contains 0, 1 or 2 (or 3 if not
                               044A                                             ;  found).
          51   B2 AF41    3C   044A             MOVZWL  TU_ABSDENS[R1],R1       ; R1 contains system density code.
     50    000003E8 8F    C4   044F             MULL    #1000,R0                ; Multiply MSCP data rate by 1000.
               50   51    C6   0456             DIVL    R1,R0                   ; Divide by density.
               50   05    C0   0459             ADDL    #5,R0                   ; Round up.
                               045C
                               045C  ;          ASSUME  MT$S_SPEED      EQ      8
          51   A9 AF      9E   045C             MOVAB   TU_ABSPEED,R1           ; R1 => Start of table.
                               0460  10$:
             81   50      91   0460             CMPB    R0,(R1)+                ; Find first entry > R0.
                  FB      1A   0463             BGTRU   10$                     ; If R0 >, loop back.
          50   FF A1      9A   0465             MOVZBL  -1(R1),R0               ; Pickup previous value.
                          05   0469             RSB                             ; Return to caller.
```

```
                    046A           .SBTTL  SET_CLEAR_SEX
                    046A
                    046A  ;+
                    046A  ; SET_CLEAR_SEX - internal subroutine to set (or not to set) the
                    046A  ;         CLEAR_Serious_EXception modifier in an MSCP command.
                    046A  ;         If the tape is NOT in Serious Exception mode, then this modifier
                    046A  ;         is routinely set on each and every command.  If the tape IS in
                    046A  ;         serious exception mode, then the modifier bit is only set if the
                    046A  ;         QIO function code modifier IO$M_CLSEREXCP is specified on this
                    046A  ;         QIO request.
                    046A  ;
                    046A  ;         Whether or not we are in Serious Exception mode is a function
                    046A  ;         of how the tape was mounted and the state of a MT$M_ENSEREXCP bit
                    046A  ;         in UCB$L_DEVDEPEND.
                    046A  ;
                    046A  ;         If the tape is MOUNTED ANSI, this implies that Serious Exception
                    046A  ;         mode is enabled.  In other words, we are in Serious Exception mode
                    046A  ;         if the volume is Mounted ANSI or if the MT$M_ENSEREXCP bit is on in
                    046A  ;         UCB$L_DEVDEPEND.  If a tape is NOT mounted ANSI (i.e. either not
                    046A  ;         mounted or mounted foreign) and MT$M_ENSEREXECP is not set then
                    046A  ;         we implicitly insert a Clear Serious Exception modifier on each
                    046A  ;         and every command.
                    046A  ;
                    046A  ; Inputs:
                    046A  ;         R2 => MSCP command buffer
                    046A  ;         R3 => UCB
                    046A  ;         R5 => CDRP
                    046A
                    046A  SET_CLEAR_SEX:
                    046A
          09    E0  046A           BBS     #IO$V_CLSEREXCP,-          ; Branch to clear if clearing serious
    0F C0 A5        046C           CDRP$W_FUNC(R5),10$        ;  exception specified.
                    046F
          02    E0  046F           BBS     #MT$V_ENSEREXCP,-          ; Branch if Serious Exception explicitly
    12 44 A3        0471           UCB$L_DEVDEPEND(R3),20$ ;  enabled.
          13    E1  0474           BBC     #DEV$V_MNT,-               ; If Tape NOT mounted, go clear serious
    05 38 A3        0476           UCB$L_DEVCHAR(R3),10$      ;  exception.
          18    E1  0479           BBC     #DEV$V_FOR,-               ; Branch around Serious Exception
    08 38 A3        047B           UCB$L_DEVCHAR(R3),20$      ;  clearing if tape MOUNTED ANSI.
                    047E
                    047E  10$:     ASSUME  MSCP$V_MD_CLSEX GE 8
          20    88  047E           BISB    #<MSCP$M_MD_CLSEX@-8>,- ; Request clearing of possible Serious
       08 A2        0480           MSCP$W_MODIFIER+1(R2)      ; Exception condition.
          01    8A  0482           BICB    #MT$M_SEREXCP,-            ; Also explicitly clear software bit.
       44 A3        0484           UCB$L_DEVDEPEND(R3)
                    0486
                05  0486  20$:     RSB                                ; Return.
```

```
0487                 .IF     DF          TU_SEQCHK
0487                 .ALIGN  LONG,0
0487      SEQ_MASK:
0487                 SEQFUNC <-                           ; SEQUENTIALFUNCTIONS
0487                         UNLOAD,-                     ;  Unload (make available + spindown)
0487                         AVAILABLE,-                  ;  Available (no spindown)
0487                         SPACERECORD,-                ;  Space Records
0487                         RECAL,-                      ;  Recalibrate (REWIND)
0487                         PACKACK,-                    ;  Pack Acknowledge
0487                         ERASETAPE,-                  ;  Erase Tape (Erase Gap)
0487                         SETCHAR,-                    ;  Set Characteristics
0487                         SETMODE,-                    ;  Set Mode
0487                         SPACEFILE,-                  ;  Space File
0487                         WRITECHECK,-                 ;  Write Check
0487                         READPBLK,-                   ;  Read  PHYSICAL Block
0487                         WRITEPBLK,-                  ;  Write PHYSICAL Block
0487                         READLBLK,-                   ;  Read  LOGICAL  Block
0487                         WRITELBLK,-                  ;  Write LOGICAL  Block
0487                         READVBLK,-                   ;  Read  VIRTUAL  Block
0487                         WRITEVBLK,-                  ;  Write VIRTUAL  Block
0487                         WRITEMARK,-                  ;  Write Tape Mark
0487                         DSE,-                        ;  Data Security Erase
0487                         REWIND,-                     ;  Rewind
0487                         REWINDOFF,-                  ;  Rewind AND Set Offline (UNLOAD)
0487                         SKIPRECORD,-                 ;  Skip Records
0487                         SKIPFILE,-                   ;  Skip Files
0487                         WRITEOF>                     ;  Write End Of File
0487                 .ENDC
```

```
                            0487          .SBTTL  AUTO_PACKACK - Perform automatic PACKACK for foreign tapes
                            0487  ;++
                            0487  ;
                            0487  ;       This code thread performs a gratuitous PACKACK for foreign mounted
                            0487  ;       tapes.  It executes whenever an I/O request finds the volume valid bit
                            0487  ;       clear, the tape at BOT, and the foreign mounted bit set.
                            0487  ;
                            0487  ;       The input CDRP is given a RSPID and a message buffer.  The message is
                            0487  ;       inititalized.  This thread is then synchronized with the server so
                            0487  ;       that this is the only thread communicating with the server.  Note:
                            0487  ;       there is an implicit synchronization with other SEQNOP threads in that
                            0487  ;       control cannot arrive here while other threads are synchronized by
                            0487  ;       SEQNOP.
                            0487  ;
                            0487  ;       Once synchronization is established, ONLINE and GET UNIT STATUS
                            0487  ;       commands are sent to the server.  This simulates an IO$_PACKACK.
                            0487  ;       If either command fails, the I/O request is completed with a volume
                            0487  ;       invalid error.  If both commands succeed, the device is marked volume
                            0487  ;       valid and BOT.  The original request is requeued at the head of the
                            0487  ;       pending I/O request queue and the SEQNOP condition is ended.  This
                            0487  ;       restarts the original I/O request before any which may have
                            0487  ;       accumulated while the automatic PACKACK was in progress.
                            0487  ;
                            0487  ;       All failures result in the unit being set MSCP AVAILABLE and the UCB
                            0487  ;       being marked volume invalid.  Before completing the original I/O
                            0487  ;       request, the error path also ends the SEQNOP condition.
                            0487  ;--
                            0487
                            0487          .ENABLE LSB
                            0487
               010B   31    0487  850$:   BRW     MSG_BUF_FAILURE         ; Branch assist.
                            048A
                            048A  AUTO_PACKACK:
                            048A
                            048A          .IIF    DF TU_SEQCHK, BSBW OVERRIDE_SEQCHK      ; Undo seq. checking.
                            048A          ALLOC_RSPID                     ; Allocate RSPID.
                            0490          ALLOC_MSG_BUF                   ; Allocate a message buffer.
              F1 50   E9    0493          BLBC    R0, 850$                ; Branch if connection broken.
                            0496          INIT_MSCP_MSG ucb=(R3)          ; Initialize message buffer.
                            0499          START_SEQNOP                    ; Synchronize with server.
                            04AF
        08 A2    09   90    04AF          MOVB    #MSCP$K_OP_ONLIN, -     ; ONLINE command.
                            04B3                  MSCP$B_OPCODE(R2)
   0A A2   2020 8F    A8    04B3          BISW    #<MSCP$M_MD_CLSEX -     ; Do exclusive ONLINE and clear serious
                            04B9                    !MSCP$M_MD_EXCLU>, -  ; exception.
                            04B9                  MSCP$W_MODIFIER(R2)
   0E A2   00E0 C3    B0    04B9          MOVW    UCB$W_UNIT_FLAGS(R3), - ; Copy UNIT flags to MSCP packet.
                            04BF                  MSCP$W_UNT_FLGS(R2)
           00D3 C3    D0    04BF          MOVL    UCB$L_MSCPDEVPARAM(R3),-; Copy Device dependent parameters to
              1C A2         04C3                  MSCP$L_DEV_PARM(R2)     ; MSCP packet.
50   44 A3   05   08  EF    04C5          EXTZV   #MT$V_DENSITY, -        ; Determine density that the user has
                            04CB                  #MT$S_DENSITY, -        ; last established for this unit
                            04CB                  UCB$L_DEVDEPEND(R3), R0 ; and put into R0.
               FF3E   30    04CB          BSBW    VMSTOMSCP_DENS          ; Convert VMS density to MSCP format.
          20 A2  51   B0    04CE          MOVW    R1, MSCP$W_FORMAT(R2)   ; Move MSCP density in R1 into packet.
       0D 0E A2  05   E1    04D2          BBC     #MSCP$V_UF_VSMSU, -     ; Test if we are suppressing variable
                            04D7                  MSCP$W_UNT_FLGS(R2), 10$; speed mode, and branch if NOT.
```

```
                  18    EF  04D7          EXTZV   #MT$V_SPEED,-             ; Extract user's speed specification
                  08        04D9                  #MT$S_SPEED,-             ; from UCB.
        50     44 A3        04DA                  UCB$L_DEVDEPEND(R3), R0
                FF50    30  04DD          BSBW    SPEEDTOMSCP
        22 A2     50    B0  04E0          MOVW    R0, MSCP$W_SPEED(R2)      ; Move MSCP speed in R0 into packet.
                            04E4    10$:  SEND_MSCP_MSG                     ; ONLIN - returns end pkt. addr. in R2.
                            04E7          ASSUME  CDRP$V_CAND EQ 0
           47 40 A5     E8  04E7          BLBS    CDRP$L_DUTUFLAGS(R5), -   ; Has operation been canceled?
                            04EB                  900$                      ; Branch if operation canceled.
                            04EB          IF_MSCP FAILURE, then=900$        ; Branch if ONLIN failed.
                            04F1
                            04F1          ; The various fields in the END PACKET are valid and the tape is
                            04F1          ; ONLINE.
                            04F1
                02A1    30  04F1          BSBW    RECORD_ONLINE             ; Move data from end message to UCB.
                            04F4
                            04F4          RESET_MSCP_MSG                    ; Setup message buf. etc. for reuse.
        08 A2     03    90  04F7          MOVB    #MSCP$K_OP_GTUNT, -       ; GET UNIT STATUS command.
                            04FB                  MSCP$B_OPCODE(R2)
                            04FB          SEND_MSCP_MSG                     ; GTUNT - returns end pkt. addr. in R2.
                            04FE          ASSUME  CDRP$V_CAND EQ 0
           30 40 A5     E8  04FE          BLBS    CDRP$L_DUTUFLAGS(R5), -   ; Has operation been canceled?
                            0502                  900$                      ; Branch if operation canceled.
                            0502          IF_MSCP FAILURE, then=900$        ; Branch if GTUNT failed.
                            0508
                0298    30  0508          BSBW    RECORD_GETUNIT_CHAR       ; Record UNIT status data in UCB.
                            050B
                            050B          ASSUME  UCB$V_VALID GE 8
        65 A3     08    88  050B          BISB    #<UCB$M_VALID @ -8>, -    ; Make unit volume valid.
                            050F                  UCB$W_STS+1(R3)
                            050F          ASSUME  MT$V_BOT GE 16
        46 A3     01    88  050F          BISB    #<MT$M_BOT @ -16>, -      ; Set beginning of tape.
                            0513                  UCB$L_DEVDEPEND+2(R3)
               FAEA'    30  0513          BSBW    DUTU$DEALLOC_ALL          ; Release all SCS resources.
     4C A3     A0 A5    0E  0516          INSQUE  CDRP$L_IOQFL(R5), -       ; Put this request at the head of
                            051B                  UCB$L_IOQFL(R3)           ; the pending I/O queue.
                            051B          END_SEQNOP                        ; End the sequential NOP state.
                        05  0531          RSB                               ; Kill this thread.
                            0532
                            0532          ; Something went wrong during auto PACKACK.  Fail the I/O request.
                            0532
                            0532   900$:  ASSUME  UCB$V_VALID GE 8
        65 A3     08    AA  0532          BICW    #<UCB$M_VALID @ -8>, -    ; Clear unit volume valid.
                            0536                  UCB$W_STS+1(R3)
     03 0A A2     07    E1  0536          BBC     #MSCP$V_SC_DUPUN, -       ; Branch around if NOT duplicate
                            053B                  MSCP$W_STATUS(R2), 940$ ; unit substatus.
               FAC2'    30  053B          BSBW    DUTU$SEND_DUPLICATE_UNIT; Notify operator of duplicate unit.
                            053E   940$:  RESET_MSCP_MSG                    ; Setup message buf. etc. for reuse.
        08 A2     08    90  0541          MOVB    #MSCP$K_OP_AVAIL, -       ; Setup available command.
                            0545                  MSCP$B_OPCODE(R2)
                            0545          SEND_MSCP_MSG                     ; AVAIL - returns end pkt. addr. in R2.
                            0548          END_SEQNOP                        ; End the sequential NOP state.
      50    0254 8F     3C  055E          MOVZWL  #SS$_VOLINV, R0           ; Set volume invalid status.
                            0563          ASSUME  CDRP$V_CAND EQ 0
           03 40 A5     E9  0563          BLBC    CDRP$L_DUTUFLAGS(R5), -   ; But, if operation was canceled,
                            0567                  950$                      ; use "aborted" status instead.
           50     2C    3C  0567          MOVZWL  #SS$_ABORT, R0
```

```
075B   31  056A  1863 950$:   BRW     FUNCTION_EXIT            ; Terminate origianl I/O request.
           056D  1864
           056D  1865         .DISABLE LSB
```

```
                          056D          .SBTTL  START I/O
                          056D  ;
                          056D  ;+
                          056D  ;
                          056D  ;  Beginning of out of line code to deal with problems that
                          056D  ;       may occur in the common STARTIO code on the next page.
                          056D
                          056D  LOCAL_DEVICE:
   55   00A8 C5     D0    056D          MOVL    UCB$L_2P_ALTUCB(R5),R5  ; R5 => local UCB.
     00000000'GF    17    0572          JMP     G^EXE$INSIOQ            ; Go hand this IRP to local driver.
                          0578
                          0578  ;
                          0578  ; Out of line code to handle Volume Invalid.
                          0578  ;
                          0578
                          0578  VOL_INVALID:
                          0578
 09 38 A3    18     E1    0578          BBC     #DEV$V_FOR, -           ; Branch if device is not foreign
                          057D                  UCB$L_DEVCHAR(R3), 10$  ; mounted.
        00B0 C3     D5    057D          TSTL    UCB$L_RECORD(R3)        ; Is device at beginning of tape?
             03     12    0581          BNEQ    10$                     ; Branch if device not at BOT.
           FF04     31    0583          BRW     AUTO_PACKACK            ; Else, go issue gratuitous PACKACK.
             08     E0    0586  10$:    BBS     #IRP$V_PHYSIO,-         ; See if PHYSICAL I/O requested.
          CA A5           0588                  CDRP$W_STS(R5),-        ;  If physical, then branch back to
             53           058A                  PHYIO_VOLINV            ;  continue even tho VOLINV.
                          058B          .IF     DF      TU_SEQCHK
                          058B          BSBW    OVERRIDE_SEQCHK         ; Override sequence checking and
                          058B                                          ;  remove sequence # from array.
                          058B          .ENDC
                          058B
   50   0254 8F     3C    058B          MOVZWL  #SS$_VOLINV,R0          ; Indicate error status.
             51     D4    0590          CLRL    R1                      ; Clear second word of I/O status.
           0733     31    0592          BRW     FUNCTION_EXIT           ; GOTO common exit.
                          0595
                          0595  ;
                          0595
                          0595  MSG_BUF_FAILURE:
                          0595
                          0595  ; We are here only if we had an allocation failure on the Message Buffer.
                          0595  ;  This implies that our CONNECTION to the MSCP server is broken.  The action
                          0595  ;  to be taken is to kill this thread of execution since we are guaranteed
                          0595  ;  that a thread exists that is currently executing that is gathering all
                          0595  ;  CDRP's associated with this CONNECTION.  So we branch to KILL_THIS_THREAD.
                          0595
          FA68'     31    0595          BRW     DUTU$KILL_THIS_THREAD   ; Branch to where we collect all active
                          0598                                          ;  CDRP's prior to re-CONNECTION.
                          0598  ;
                          0598  ;  End of out of line code
                          0598  ;-
```

```
                                 0598  TU_STARTIO:
                                 0598          ASSUME  UCB$V_BSY GE 8
            65 A5    01    8A    0598          BICB    #<UCB$M_BSY @ -8>, -     ; Undo bit setting so that multiple
                                 059C                  UCB$W_STS+1(R5)          ; IRP's can be started.
                                 059C
                                 059C  ; If this UCB indicates that the device is a local (non-MSCP) device that
                                 059C  ; has also been made available to us via 1) dual porting and 2) an MSCP
                                 059C  ; Server on the node to which it is dual ported, then shunt this IRP to
                                 059C  ; the local driver.
                                 059C
                     03    E0    059C          BBS     #DEV$V_CDP,-             ; This bit, if clear indicates that
                  3C A5          059E                  UCB$L_DEVCHAR2(R5),-     ;  the above condition is NOT true,
                     CC          05A0                  LOCAL_DEVICE             ;  so branch out of line if set.
            50    60 A3    9E    05A1          MOVAB   -CDRP$L_IOQFL(R3),R0     ; Get address of CDRP portion of IRP.
                                 05A5
                                 05A5          ASSUME  CDRP$B_CD_TYPE EQ CDRP$W_CDRPSIZE+2
                                 05A5          ASSUME  CDRP$B_FIPL    EQ CDRP$W_CDRPSIZE+3
   08 A0    0839FFA0 8F    D0    05A5          MOVL    #< <IPL$_SCS@24> -       ; Initialize CDRP size, type and fork
                                 05AD                   ! <DYN$C_CDRP@16> -     ; IPL fields.
                                 05AD                   ! <CDRP$C_IOQFL&^xFFFF> >, -
                                 05AD                  CDRP$W_CDRPSIZE(R0)
                                 05AD
                                 05AD          ASSUME  CDRP$L_RSPID    EQ      CDRP$L_MSG_BUF+4
                  1C A0    7C    05AD          CLRQ    CDRP$L_MSG_BUF(R0)       ; Prevent spurious DEALLOC_MSG_BUF and
                                 05B0                                           ;  also spurious DEALLOC_RSPID.
                  2C A0    D4    05B0          CLRL    CDRP$L_LBUFH_AD(R0)      ; Prevent spurious UNMAP.
                  56 A5    9E    05B3          MOVAB   UCB$W_RWAITCNT(R5),-     ; Point CDRP field to UCB field.
                  28 A0          05B6                  CDRP$L_RWCPTR(R0)
                  40 A0    D4    05B8          CLRL    CDRP$L_DUTUFLAGS(R0)     ; Initialize class driver flags.
                  56 A5    B5    05BB          TSTW    UCB$W_RWAITCNT(R5)       ; See if any IRP's currently waiting
                                 05BE                                           ;  for resources.
                     05    13    05BE          BEQL    TU_REAL_STARTIO          ; EQL implies NO, so GOTO real STARTIO.
                     63    0E    05C0          INSQUE  IRP$L_IOQFL(R3),-        ; To force sequential submission of commands
                  50 B5          05C2                  @UCB$L_IOQBL(R5)         ;  to intelligent controller, we force
                                 05C4                                           ;  IRP's to be queued up here if any
                                 05C4                                           ;  previous request is possibly hungup
                                 05C4                                           ;  waiting for resources between the
                                 05C4                                           ;  beginning of STARTIO and the SEND_MSG_BUF
                           05    05C4          RSB                              ; Return to caller (QIO system service)
                                 05C5
                                 05C5  TU_REAL_STARTIO:
                                 05C5
                                 05C5          .IF     DF      TU_TRACE
                                 05C5          BSBW    TRACE_IRP                ; Trace IRP.
                                 05C5          MOVAB   -CDRP$L_IOQFL(R3),R0     ; Refresh R0=CDRP if tracing.
                                 05C5          .ENDC
                                 05C5
                  53    55 D0    05C5          MOVL    R5,R3                    ; Let R3 => UCB.
                  55    50 D0    05C8          MOVL    R0,R5                    ; R5 => CDRP.
                                 05CB
                                 05CB          .IF     DF      TU_SEQCHK
                                 05CB          EXTZV   #IRP$V_FCODE,-           ; Extract I/O function code.
                                 05CB                  #IRP$S_FCODE,-
                                 05CB                  CDRP$W_FUNC(R5),R1
                                 05CB          BBC     R1,SEQ_MASK,TU_RESTARTIO; If non-Sequential I/O branch around.
                                 05CB          EXTZV   #0,-                     ; Extract six bit index into array of
                                 05CB                  #6,-                     ;  IRP sequence number slots.  R1 =
                     05CB  1974                 UCB$B_TU_NEWINX(R3),R1  ;  index of next available slot.
                     05CB  1975         INCB    UCB$B_TU_NEWINX(R3)     ; Increment index.
                     05CB  1976         MOVL    CDRP$L_SEQNUM(R5),-     ; Copy sequnce number of this IRP to
                     05CB  1977                 UCB$L_TU_SEQARY(R3)[R1] ;  circular ring slot.
                     05CB  1978         .ENDC
                     05CB  1979
                     05CB  1980 TU_RESTARTIO:                           ; Label where we RESTART CDRP's after
                     05CB  1981                                         ;  virtual circuit re-CONNECTION.
                     05CB  1982
         00C8 C3  D0 05CB  1983         MOVL    UCB$L_CDT(R3),-         ; Place CDT pointer into CDRP for handy
           24 A5     05CF  1984                 CDRP$L_CDT(R5)          ;  reference by SCS routines. Note we
                     05D1  1985                                         ;  do this after label TU_RESTARTIO so
                     05D1  1986                                         ;  that it is refreshed upon restart.
     54  0084 C3  D0 05D1  1987         MOVL    UCB$L_PDT(R3),R4        ; R4 => port's PDT.
                     05D6  1988
    03 64 A3  08  E0 05D6  1989         BBS     #UCB$V_VALID, -         ; Branch if unit is volume valid.
                     05DB  1990                 UCB$W_STS(R3), PHYIO_VOLINV
            FF9A  31 05DB  1991         BRW     VOL_INVALID             ; Else, branch to out of line
                     05DE  1992                                         ; volume invalid processing.
                     05DE  1993
                     05DE  1994 PHYIO_VOLINV:
                     05DE  1995         ALLOC_RSPID                     ; ALLOCate a ReSPonse ID.
                     05E4  1996         ALLOC_MSG_BUF                   ; Allocate an MSCP buffer (and also
                     05E7  1997                                         ;  allocate a unit of flow control).
           AB 50  E9 05E7  1998         BLBC    R0,MSG_BUF_FAILURE      ; If failure, branch out of line.
                     05EA  1999
                     05EA  2000 ; Here a little common MSCP packet initialization.
                     05EA  2001
          50  52  D0 05EA  2002         MOVL    R2, R0                  ; Copy message buffer address.
                     05ED  2003         .REPEAT MSCP$K_MXCMDLEN / 8
                     05ED  2004         CLRQ    (R0)+                   ; Zero entire message buffer.
              80  7C 05ED  2005         .ENDR
              80  D4 05F5  2006         .IIF    NE MSCP$K_MXCMDLEN & 4, CLRL (R0)+
                     05F7  2007         .IIF    NE MSCP$K_MXCMDLEN & 2, CLRW :R0)+
                     05F7  2008         .IIF    NE MSCP$K_MXCMDLEN & 1, CLRB (R0)+
                     05F7  2009
           20 A5  D0 05F7  2010         MOVL    CDRP$L_RSPID(R5),-      ; Use RSPID as command reference
              62     05FA  2011                 MSCP$L_CMD_REF(R2)      ;  number for all commands.
                     05FB  2012
         00D4 C3  B0 05FB  2013         MOVW    UCB$W_MSCPUNIT(R3),-    ; Indicate UNIT number in MSCP
           04 A2     05FF  2014                 MSCP$W_UNIT(R2)         ;  packet.
                     0601  2015
                     0601  2016 TU_BEGIN_IVCMD:
                     0601  2017 TU_REDO_IO:
                     0601  2018
            FE66  30 0601  2019         BSBW    SET_CLEAR_SEX           ; Go set state of Clear Serious EXception.
              0F  E1 0604  2020         BBC     #IO$V_INHRETRY,-        ; Branch around if NOT inhibiting RETRY.
        04 C0 A5     0606  2021                 CDRP$W_FUNC(R5),30$
                     0609  2022         ASSUME  MSCP$V_MD_SEREC GE 8    ; Else, set the suppress error
       0B A2  01  88 0609  2023         BISB    #<MSCP$M_MD_SEREC@-8>, -; modifier.
                     060D  2024                 MSCP$W_MODIFIER+1(R2)
                     060D  2025 30$:
              00  EF 060D  2026         EXTZV   #IRP$V_FCODE,-          ; Extract I/O function code.
              06     060F  2027                 #IRP$S_FCODE,-
       51  C0 A5     0610  2028                 CDRP$W_FUNC(R5),R1
                     0613  2029
                     0613  2030         DISPATCH R1, type=B, prefix=IO$_, < -           ; Dispatch to correct
```

```
                        0613          <NOP,           START_NOP>, -           ; function processing.
                        0613          <PACKACK,       START_PACKACK>, -
                        0613          <UNLOAD,        START_UNLOAD>, -
                        0613          <AVAILABLE,     START_AVAILABLE>, -
                        0613          <REWIND,        START_REWIND>, -
                        0613          <REWINDOFF,     START_REWINDOFF>, -
                        0613          <READPBLK,      START_READPBLK>, -
                        0613          <WRITECHECK,    START_WRITECHECK>, -
                        0613          <WRITEPBLK,     START_WRITEPBLK>, -
                        0613          <WRITEMARK,     START_WRITEMARK>, -
                        0613          <WRITEOF,       START_WRITEOF>, -
                        0613          <SPACEFILE,     START_SPACEFILE>, -
                        0613          <SKIPFILE,      START_SKIPFILE>, -
                        0613          <SPACERECORD,   START_SPACERECORD>, -
                        0613          <SKIPRECORD,    START_SKIPRECORD>, -
                        0613          <RECAL,         START_RECAL>, -
                        0613          <ERASETAPE,     START_ERASETAPE>, -
                        0613          <DSE,           START_DSE>, -
                        0613          <SENSECHAR,     START_SENSECHAR>, -
                        0613          <SENSEMODE,     START_SENSEMODE>, -
                        0613          <SETCHAR,       START_SETCHAR>, -
                        0613          <SETMODE,       START_SETMODE> -
                        0613          >
                        0669
                        0669     ; Function code is not legal.
                        0669
             F994'  30  0669     BSBW    DUTU$RESTORE_CREDIT     ; Restore allocated send credit.
    50  00F4 8F     3C  066C     MOVZWL  #SS$_ILLIOFUNC,R0
               51   D4  0671     CLRL    R1
             0652   31  0673     BRW     FUNCTION_EXIT           ; Branch to exit I/O function.
```

```
                  0676          .SBTTL  START_NOP
                  0676  ; START_NOP - Prepare an MSCP packet to do a GET UNIT STATUS command.
                  0676  ;
                  0676  ; INPUTS:
                  0676  ;       R2 => MSCP buffer
                  0676  ;       R3 => UCB
                  0676  ;       R4 => PDT
                  0676  ;       R5 => CDRP
                  0676  ;
                  0676  ;       MSCP packet is zero except for MSCP$L_CMD_REF and MSCP$W_UNIT fields.
                  0676  ;
                  0676
                  0676  START_NOP:
       03    90   0676          MOVB    #MSCP$K_OP_GTUNT,-      ; Transfer GET UNIT STATUS opcode
    08 A2         0678                  MSCP$B_OPCODE(R2)       ;  to packet.
                  067A          ASSUME  MSCP$V_MD_CLSEX GE 8
       20    8A   067A          BICB    #<MSCP$M_MD_CLSEX@-8>,- ; The clear serious execption modifier
    08 A2         067C                  MSCP$W_MODIFIER+1(R2)   ; is illegal on get unit status cmds.
                  067E
                  067E          IF_IVCMD then=NOP_IVCMD_END     ; Branch if invalid command processing.
                  0682
                  0682          SEND_MSCP_MSG                   ; Send message to remote MSCP server.
                  0685
                  0685          DO_ACTION       NONTRANSFER     ; Decode MSCP end status.
                  0688          ACTION_ENTRY    SUCC,  SS$_NORMAL,      NOP_SUCC
                  068D          ACTION_ENTRY    OFFLN, SS$_DEVOFFLINE,  NOP_OFFLINE
                  0692          ACTION_ENTRY    AVLBL, SS$_MEDOFL,      NOP_AVAIL
                  0697          ACTION_ENTRY    DRIVE, SS$_DRVERR,      NOP_DRVERR
                  069C          ACTION_ENTRY    CNTLR, SS$_CTRLERR,     NOP_CTRLERR
                  06A1          ACTION_ENTRY    ICMD,  SS$_CTRLERR,     NOP_IVCMD
                  06A6          ACTION_ENTRY    END_TABLE
                  06A8
     09CE    31   06AB          BRW     INVALID_STS             ; Unexpected MSCP end status.
                  06AB
                  06AB  NOP_IVCMD:
                  06AB          IVCMD_BEGIN                     ; Begin invalid command processing.
     FF50    31   06AE          BRW     TU_BEGIN_IVCMD          ; Replicate building MSCP command.
                  06B1  NOP_IVCMD_END:
                  06B1          IVCMD_END                       ; Complete invalid command processing.
                  06B3  ; ----- BRB     NOP_SUCC                ; Fall through to complete command.
                  06B3
                  06B3
                  06B3  NOP_SUCC:
                  06B3  NOP_OFFLINE:
                  06B3  NOP_AVAIL:
                  06B3  NOP_CTRLERR:
                  06B3  NOP_DRVERR:
                  06B3  ;NOP_END:
       51    D4   06B3          CLRL    R1                      ; Clear for I/O status block.
     0610    31   06B5          BRW     FUNCTION_EXIT           ; Branch to common exit.
```

```
                       06B8          .SBTTL  START_PACKACK
                       06B8
                       06B8  ; START_PACKACK - Prepare an MSCP packet to do an ONLINE command.
                       06B8  ;
                       06B8  ; INPUTS:
                       06B8  ;       R2 => MSCP buffer
                       06B8  ;       R3 => UCB
                       06B8  ;       R4 => PDT
                       06B8  ;       R5 => CDRP
                       06B8  ;
                       06B8  ;       MSCP packet is zero except for MSCP$L_CMD_REF and MSCP$W_UNIT fields.
                       06B8  ;
                       06B8
                       06B8  START_PACKACK:
                       06B8
              09   90  06B8          MOVB    #MSCP$K_OP_ONLIN,-      ; Transfer ONLINE opcode
            08 A2      06BA                  MSCP$B_OPCODE(R2)       ;  to packet.
                       06BC
    50    00BC C3  D0  06BC          MOVL    UCB$L_CDDB(R3), R0      ; Get CDDB address.
  04 28 A0     02  E1  06C1          BBC     #MSCP$V_CF_MLTHS, -     ; Branch if not a multi-host server.
                       06C6                  CDDB$W_CNTRLFLGS(R0), 20$
              20   A8  06C6          BISW    #MSCP$M_MD_EXCLU,-      ; Do exclusive ONLINE.
            0A A2      06C8                  MSCP$W_MODIFIER(R2)
                       06CA
 0E A2    00E0 C3  B0  06CA  20$:    MOVW    UCB$W_UNIT_FLAGS(R3), - ; Copy unit flags to MSCP packet.
                       06D0                  MSCP$W_UNT_FLGS(R2)
                       06D0
          00D8 C3  D0  06D0          MOVL    UCB$L_MSCPDEVPARAM(R3),-; Copy Device dependent parameters to
            1C A2      06D4                  MSCP$L_DEV_PARM(R2)     ;  MSCP packet.
                       06D6
              08   EF  06D6          EXTZV   #MT$V_DENSITY,-         ; Determine density that the user has
              05       06D8                  #MT$S_DENSITY,-         ;  last established for this unit
    50      44 A3      06D9                  UCB$L_DEVDEPEND(R3),R0  ;  and put into R0.
            FD2D   30  06DC          BSBW    VMSTOMSCP_DENS          ; Convert VMS density to MSCP format.
  20 A2        51  B0  06DF          MOVW    R1,MSCP$W_FORMAT(R2)    ; Move MSCP density in R1 into packet.
                       06E3
                       06E3          IF_IVCMD then=PACKACK_IVCMD_END ; Branch if invalid command processing.
                       06E7
                       06E7          SEND_MSCP_MSG                   ; Send message to remote MSCP server.
                       06EA
                       06EA          ASSUME  UCB$V_VALID GE 8
    65 A3      08  8A  06EA          BICB    #<UCB$M_VALID @ -8>, -  ; Initialize software volume invalid.
                       06EE                  UCB$W_STS+1(R3)
                       06EE
                       06EE          DO_ACTION       NONTRANSFER     ; Decode MSCP end status.
                       06F1          ACTION_ENTRY    SUCC,  SS$_NORMAL,      PACKACK_SUCC
                       06F6          ACTION_ENTRY    OFFLN, SS$_MEDOFL,      PACKACK_OFFLINE
                       06FB          ACTION_ENTRY    ABRTD, SS$_ABORT,       END_PACKACK
                       0700          ACTION_ENTRY    DRIVE, SS$_DRVERR,      END_PACKACK
                       0705          ACTION_ENTRY    FMTER, SS$_CTRLERR,     END_PACKACK
                       070A          ACTION_ENTRY    CNTLR, SS$_CTRLERR,     END_PACKACK
                       070F          ACTION_ENTRY    ICMD,  SS$_CTRLERR,     PACKACK_IVCMD
                       0714          ACTION_ENTRY    END_TABLE
                       0716
            0960   31  0716          BRW     INVALID_STS             ; Unexpected MSCP end status.
                       0719
                       0719
```

```
                    0719  PACKACK_SUCC:                              ; Action routine for MSCP$K_ST_SUCC.
                    0719
                    0719          ASSUME  CDRP$V_CAND EQ 0
   24 40 A5    E8   0719          BLBS    CDRP$L_DUTUFLAGS(R5), -   ; Was I/O request canceled?
                    071D                  890$                      ; Branch if request was canceled.
         08    E0   071D          BBS     #MSCP$V_SC_ALONL,-        ; Branch around clearing of TU_RECORD
   0C 0A A2         071F                  MSCP$W_STATUS(R2),10$     ;   if REDUNDANT ONLINE.
    0080 C3    D4   0722          CLRL    UCB$L_RECORD(R3)          ; Successful exclusive ONLINE rewinds
                    0726          ASSUME  MT$V_BOT  GE 16
                    0726          ASSUME  MT$V_EOF  GE 16
                    0726          ASSUME  MT$V_EOT  GE 16
                    0726          ASSUME  MT$V_LOST GE 16
  46 A3   16   8A   0726          BICB    #<<MT$M_EOF ! MT$M_EOT -; Clear position sensitive DEVDEPEND
                    072A                   ! MT$M_LOST> @ -16>, - ; bits.
                    072A                  UCB$L_DEVDEPEND+2(R3)
  46 A3   01   88   072A          BISB    #<MT$M_BOT @ -16>, -      ; Set BOT DEVDEPEND pos:tion bit.
                    072E                  UCB$L_DEVDEPEND+2(R3)
                    072E  10$:
       0064    30   072E          BSBW    RECORD_ONLINE             ; Record ONLINE data in UCB.
                    0731
                    0731  ; Here having done an ONLINE we proceed to do a GET UNIT STATUS.
                    0731
                    0731          RESET_MSCP_MSG                    ; Setup message buf. etc. for reuse.
         03    90   0734          MOVB    #MSCP$K_OP_GTUNT,-        ; Opcode is for GET UNIT STATUS.
      08 A2         0736                  MSCP$B_OPCODE(R2)
                    0738          SEND_MSCP_MSG                     ; Send message to remote MSCP server.
                    073B
                    073B          IF_MSCP SUCCESS, then=PACKACK_GTUNT_SUCC  ; Branch if GTUNT successful.
                    0741          ASSUME  CDRP$V_CAND EQ 0
   3A 40 A5    E8   0741  890$:   BLBS    CDRP$L_DUTUFLAGS(R5), -   ; Was I/O request canceled?
                    0745                  PACKACK_CANCEL            ; Branch if request was canceled.
                    0745          RESET_MSCP_MSG                    ; Setup message buf. etc. for reuse.
       FEB6    31   0748          BRW     TU_REDO_IO                ; Go try again.
                    074B
                    074B  PACKACK_GTUNT_SUCC:
                    074B
         56    10   074B          BSBB    RECORD_GETUNIT_CHAR       ; Record unit status data in UCB.
                    074D
     50  01    3C   074D          MOVZWL  #SS$_NORMAL, R0           ; Set success IOSB status.
         3C    11   0750          BRB     VALID_PACKACK             ; And branch around to success.
                    0752
                    0752  PACKACK_IVCMD:
                    0752          IVCMD_BEGIN                       ; Begin invalid command processing.
       FEA9    31   0755          BRW     TU_BEGIN_IVCMD            ; Repeat commands that formed MSCP cmd.
                    0758  PACKACK_IVCMD_END:
                    0758          IVCMD_END                         ; Complete invalid command processing.
         36    11   075A          BRB     END_PACKACK               ; Branch around to end.
                    075C
                    075C  PACKACK_OFFLINE:
                    075C
         07    E1   075C          BBC     #MSCP$V_SC_DUPUN,-        ; Branch around if NOT duplicate
   12 0A A2         075E                  MSCP$W_STATUS(R2),20$     ;  unit substatus.
         55    DD   0761          PUSHL   R5                        ; Save R5.
   55    53    D0   0763          MOVL    R3,R5                     ; R5 => UCB for subroutine.
      F897'    30   0766          BSBW    DUTU$SEND_DUPLICATE_UNIT; Send a message to the operator.
         55  8ED0   0769          POPL    R5                        ; Restore R5.
50   21C4 8F   3C   076C          MOVZWL  #SS$_DUPUNIT,R0           ; Return final status.
```

```
                 1F   11 0771          BRB     END_PACKACK               ; Branch around.
                         0773 20$:
                 06   E1 0773          BBC     #MSCP$V_SC_INOPR,-        ; Branch around if NOT unit inoperative
              0A A2      0775                  MSCP$W_STATUS(R2),-       ;  substatus.
                 1A      0777                  END_PACKACK
      50   008C  8F   3C 0778          MOVZWL  #SS$_DRVERR,R0            ; Return final status.
                 13   11 077D          BRB     END_PACKACK               ; Branch around.
                         077F
                         077F PACKACK_CANCEL:
                         077F
                         077F          RESET_MSCP_MSG                    ; Ready message for a new MSCP command.
         08 A2   08   90 0782          MOVB    #MSCP$K_OP_AVAIL, -       ; Undo online with available command.
                         0786                  MSCP$B_OPCODE(R2)
                         0786          SEND_MSCP_MSG                     ; Sent AVAILABLE to the server.
            50   2C   3C 0789          MOVZWL  #SS$_ABORT, R0            ; Signal request was canceled.
                 04   11 078C          BRB     END_PACKACK               ; Exit function.
                         078E
                         078E VALID_PACKACK:
                         078E
                         078E          ASSUME  UCB$V_VALID GE 8
         65 A3   08   88 078E          BISB    #<UCB$M_VALID @ -8>, -    ; Set software volume valid.
                         0792                  UCB$W_STS+1(R3)
                         0792 END_PACKACK:
               0533   31 0792          BRW     FUNCTION_EXIT
```

```
                                 0795  2253          .SBTTL  PACKACK Support Routines
                                 0795  2254
                                 0795  2255  ;+
                                 0795  2256  ; RECORD_ONLINE - copy data from ONLINE END MESSAGE to UCB.
                                 0795  2257  ; RECORD_SETUNIT_CHAR - copy data from SET UNIT CHAR End Message to UCB.
                                 0795  2258  ; RECORD_GETUNIT_CHAR - copy data from GET UNIT CHAR End Message to UCB.
                                 0795  2259  ;
                                 0795  2260  ; Inputs:
                                 0795  2261  ;       R2 => End Message
                                 0795  2262  ;       R3 => UCB
                                 0795  2263  ;
                                 0795  2264  ; Outputs:
                                 0795  2265  ;       R1 corrupted.
                                 0795  2266  ;       All other registers preserved.
                                 0795  2267  ;
                                 0795  2268  ;       UCB fields set
                                 0795  2269  ;-
                                 0795  2270
                                 0795  2271  RECORD_ONLINE:
                                 0795  2272  RECORD_SETUNIT_CHAR:
                                 0795  2273
                      24 A2   D0 0795  2274          MOVL    MSCP$L_MAXWTREC(R2),-   ; Copy maximum recommended write
                    00EC C3      0798  2275                  UCB$L_TU_MAXWRCNT(R3)   ;  record size to UCB.
                      28 A2   B0 079B  2276          MOVW    MSCP$W_NOISEREC(R2),-   ; Copy size of noise records to UCB.
                    00F4 C3      079E  2277                  UCB$W_TU_NOISE(R3)
                         07   11 07A1  2278          BRB     RECORD_COMMON           ; Join common "record" processing.
                                 07A3  2279
                                 07A3  2280  RECORD_GETUNIT_CHAR:
                                 07A3  2281
                                 07A3  2282          ASSUME  MT$V_SUP_NRZI EQ 21
                                 07A3  2283          ASSUME  MSCP$V_TF_800 EQ  0
                                 07A3  2284          ASSUME  MT$V_SUP_PE   EQ 22
                                 07A3  2285          ASSUME  MSCP$V_TF_PE  EQ  1
                                 07A3  2286          ASSUME  MT$V_SUP_GCR  EQ 23
                                 07A3  2287          ASSUME  MSCP$V_TF_GCR EQ  2
   44 A3    03    15  24 A2   F0 07A3  2288          INSV    MSCP$W_FORMENU(R2), -   ; Copy supported tape densities to
                                 07AA  2289                  #MT$V_SUP_NRZI, #3, -   ; DEVDEPEND.
                                 07AA  2290                  UCB$L_DEVDEPEND(R3)
                                 07AA  2291
                                 07AA  2292  RECORD_COMMON:
                                 07AA  2293
                         50   DD 07AA  2294          PUSHL   R0                      ; Save R0.
                      14 A2   7D 07AC  2295          MOVQ    MSCP$Q_UNIT_ID(R2),-    ; In the event of success, copy unit
                    00CC C3      07AF  2296                  UCB$Q_UNIT_ID(R3)       ;  characteristics data to UCB.
                      1C A2   D0 07B2  2297          MOVL    MSCP$L_MEDIA_ID(R2),-   ; Starting with the UNIT ID, followed
                    008C C3      07B5  2298                  UCB$L_MEDIA_ID(R3)      ;  by the media identifier and
                      F845'   30 07B8  2299          BSBW    DUTU$GET_DEVTYPE        ;  device type.
                                 07BB  2300
                    1F00 8F   AA 07BB  2301          BICW    #MT$M_DENSITY,-         ; Clear density field in DEVDEPEND.
                      44 A3      07BF  2302                  UCB$L_DEVDEPEND(R3)
                                 07C1  2303
             50       20 A2   3C 07C1  2304          MOVZWL  MSCP$W_FORMAT(R2),R0    ; Pickup MSCP density code.
                       FC5D   30 07C5  2305          BSBW    MSCPTOVMS_DENS          ; Convert to VMS format.
                         50   F0 07C8  2306          INSV    R0,-                    ; Insert system density code into
                                 07CA  2307                  #MT$V_DENSITY,-         ;  DEVDEPEND.
                   05    08      07CA  2308                  #MT$S_DENSITY,-
                      44 A3      07CC  2309                  UCB$L_DEVDEPEND(R3)
```

```
                        07CE
       0E A2     B0     07CE               MOVW     MSCP$W_UNT_FLGS(R2),-      ; Copy new unit flags from end packet.
     00E0 C3            07D1                        UCB$W_UNIT_FLAGS(R3)
       22 A2     B0     07D4               MOVW     MSCP$W_SPEED(R2),-         ; Copy speed to UCB.
     00F2 C3            07D7                        UCB$W_TU_SPEED(R3)
       20 A2     B0     07DA               MOVW     MSCP$W_FORMAT(R2),-        ; Copy format to UCB.
     00F0 C3            07DD                        UCB$W_TU_FORMAT(R3)
          05     E0     07E0               BBS      #MSCP$V_UF_VSMSU,-         ; Branch if suppressing Variable speed
    04 0E A2            07E2                        MSCP$W_UNT_FLGS(R2),10$    ;  mode.
                        07E5    ;          ASSUME   MT$K_SPEED_DEF  EQ         0
          50     D4     07E5               CLRL     R0                         ; R0 = default speed.
          0B     11     07E7               BRB      20$                        ; Branch around.
                        07E9    10$:
50     22 A2     3C     07E9               MOVZWL   MSCP$W_SPEED(R2),R0        ; Get speed of unit.
51     20 A2     3C     07ED               MOVZWL   MSCP$W_FORMAT(R2),R1       ; And density.
        FC51     30     07F1               BSBW     MSCPTOSPEED                ; Convert Speed to VMS value.
                        07F4    20$:
          50     F0     07F4               INSV     R0,-                       ; Insert VMS speed value into UCB.
                        07F6                        #MT$V_SPEED,-
      08  18            07F6                        #MT$S_SPEED,-
       44 A3            07F8                        UCB$L_DEVDEPEND(R3)
                        07FA               ASSUME   MSCP$V_UF_WRTPH GE 8
                        07FA               ASSUME   MSCP$V_UF_WRTPS GE 8
                        07FA               ASSUME   MT$V_HWL GE 16
                        07FA               ASSUME   UCB$V_MSCP_WRTP GE 8
46 A3     08     8A     07FA               BICB     #<MT$M_HWL @ -16>, -       ; Assume device is not hardware write
                        07FE                        UCB$L_DEVDEPEND+2(R3)      ; locked.
          20     8A     07FE               BICB     #<UCB$M_MSCP_WRTP@-8>,-    ; Ditto for class driver write
       69 A3            0800                        UCB$W_DEVSTS+1(R3)         ; protect flag.
          93            0802               BITB     #<<MSCP$M_UF_WRTPH -       ; Is the unit hardware or
                        0803                         !MSCP$M_UF_WRTPS>@-8>,-; software write protected?
0F A2     30            0803                        MSCP$W_UNT_FLGS+1(R2)
          08     13     0806               BEQL     50$                        ; Branch if not write protected.
46 A3     08     88     0808               BISB     #<MT$M_HWL @ -16>, -       ; Else, set the hardware write
                        080C                        UCB$L_DEVDEPEND+2(R3)      ; locked bit in DEVDEPEND.
          20     88     080C               BISB     #<UCB$M_MSCP_WRTP@-8>,-    ; Set class driver write
       69 A3            080E                        UCB$W_DEVSTS+1(R3)         ; protect flag too.
                        0810
          50 8ED0       0810    50$:       POPL     R0                         ; Restore R0.
                 05     0813               RSB                                 ; Return to caller.
```

```
                        0814            .SBTTL  START_UNLOAD and START_AVAILABLE
                        0814
                        0814    ; START_AVAILABLE - Prepare an MSCP packet to do an AVAILABLE command without
                        0814    ;         the spindown modifier.
                        0814    ;
                        0814    ; START_UNLOAD - Prepare an MSCP packet to do an AVAILABLE command with
                        0814    ;         spindown specified.
                        0814    ;
                        0814    ; INPUTS:
                        0814    ;         R2 => MSCP buffer
                        0814    ;         R3 => UCB
                        0814    ;         R4 => PDT
                        0814    ;         R5 => CDRP
                        0814    ;
                        0814    ;         MSCP packet is zero except for MSCP$L_CMD_REF and MSCP$W_UNIT fields.
                        0814    ;
                        0814
                        0814 START_REWINDOFF:
                        0814 START_UNLOAD:
                        0814
             10    A8   0814            BISW    #MSCP$M_MD_UNLOD,-      ; Specify the UNLOAD bit in the
          0A A2         0816                    MSCP$W_MODIFIER(R2)     ;  modifier word.
                        0818
                        0818 START_AVAILABLE:
                        0818
             08    90   0818            MOVB    #MSCP$K_OP_AVAIL,-      ; Transfer AVAILABLE opcode
          08 A2         081A                    MSCP$B_OPCODE(R2)       ;  to packet.
                        081C
                        081C            IF_IVCMD then=AVAIL_IVCMD_END   ; Branch if invalid command processing.
                        0820
                        0820            SEND_MSCP_MSG                   ; Send message to remote MSCP server.
                        0823
                        0823            ASSUME  UCB$V_VALID GE 8
   65 A3     08    8A   0823            BICB    #<UCB$M_VALID @ -8>, -  ; Initialize software volume invalid.
                        0827                    UCB$W_STS+1(R3)
                        0827
                        0827            DO_ACTION          NONTRANSFER     ; Decode MSCP end status.
                        082A            ACTION_ENTRY    SUCC,  SS$_NORMAL,      AVAILABLE_SUCC
                        082F            ACTION_ENTRY    AVLBL, SS$_NORMAL,      AVAILABLE_SUCC
                        0834            ACTION_ENTRY    PRESE, SS$_SERIOUSEXCP, AVAILABLE_SEREX
                        0839            ACTION_ENTRY    OFFLN, SS$_MEDOFL,      AVAILABLE_MEDOFL
                        083E            ACTION_ENTRY    ABRTD, SS$_ABORT,       AVAILABLE_ABORT
                        0843            ACTION_ENTRY    DRIVE, SS$_DRVERR,      AVAILABLE_DRVERR
                        0848            ACTION_ENTRY    CNTLR, SS$_CTRLERR,     AVAILABLE_CTRLERR
                        084D            ACTION_ENTRY    ICMD,  SS$_CTRLERR,     AVAIL_IVCMD
                        0852            ACTION_ENTRY    END_TABLE
                        0854
           0822    31   0854            BRW     INVALID_STS             ; Unexpected MSCP end status.
                        0857
                        0857 AVAIL_IVCMD:
                        0857            IVCMD_BEGIN                     ; Begin invalid command processing.
           FDA4    31   085A            BRW     TU_BEGIN_IVCMD          ; Repeat building the MSCP command.
                        085D AVAIL_IVCMD_END:
                        085D            IVCMD_END                       ; Complete invalid command processing.
                        085F ; ----- BRB     AVAILABLE_SUCC          ; Fall through to complete operation.
                        085F
                        085F
```

```
                     085F   2408 AVAILABLE_SUCC:                                    ; Action routine for MSCP$K_ST_SUCC.
                     085F   2409 AVAILABLE_MEDOFL:                                  ; Action routine for MSCP$K_ST_MEDOFL.
                     085F   2410 AVAILABLE_ABORT:                                   ; Action routine for MSCP$K_ST_ABORT.
                     085F   2411 AVAILABLE_DRVERR:                                  ; Action routine for MSCP$K_ST_DRVERR.
                     085F   2412 AVAILABLE_CTRLERR:                                 ; Action routine for MSCP$K_ST_CNTLR.
        04     CA    085F   2413             BICL    #MT$M_ENSEREXCP,-              ; Clear Serious Exception mode on
     44 A3           0861   2414                     UCB$L_DEVDEPEND(R3)            ;  becoming available.
        00     F0    0863   2415             INSV    #MT$K_SPEED_DEF,-              ; Reset Speed to default.
        18           0865   2416                     #MT$V_SPEED,-
        08           0866   2417                     #MT$S_SPEED,-
     44 A3           0867   2418                     UCB$L_DEVDEPEND(R3)
        20     AA    0869   2419             BICW    #MSCP$M_UF_VSMSU,-             ; Also reset bit.
   00E0 C3           086B   2420                     UCB$W_UNIT_FLAGS(R3)
   00B0 C3     D4    086E   2421             CLRL    UCB$L_RECORD(R3)               ; Clear tape position counter.
                     0872   2422             ASSUME  MT$V_BOT   GE 16
                     0872   2423             ASSUME  MT$V_EOF   GE 16
                     0872   2424             ASSUME  MT$V_EOT   GE 16
                     0872   2425             ASSUME  MT$V_HWL   GE 16
                     0872   2426             ASSUME  MT$V_LOST  GE 16
  46 A3     1E     8A    0872   2427             BICB    #<<MT$M_EOF ! MT$M_EOT -      ; Clear position sensitive writelock
                     0876   2428                      ! MT$M_HWL ! MT$M_LOST> - ; DEVDEPEND bits.
                     0876   2429                      @ -16>, UCB$L_DEVDEPEND+2(R3)
  46 A3     01     88    0876   2430             BISB    #<MT$M_BOT @ -16>, -           ; Set BOT DEVDEPEND position bit.
                     087A   2431                     UCB$L_DEVDEPEND+2(R3)
                     087A   2432             ASSUME  UCB$V_MSCP_WRTP GE 8
        20     8A    087A   2433             BICB    #<UCB$M_MSCP_WRTP@-8>,- ; Clear class driver write
     69 A3           087C   2434                     UCB$W_DEVSTS+1(R3)             ; protect flag.
                     087E   2435 AVAILABLE_SEREX:
      0447     31    087E   2436             BRW     FUNCTION_EXIT
```

```
                      0881          .SBTTL  Start WRITEOF, WRITEMARK, ERASETAPE, and DSE.
                      0881
                      0881  ; START_WRITEMARK - Prepare an MSCP packet to do a WRITE TAPE MARK command.
                      0881  ; START_ERASETAPE - Prepare an MSCP packet to do an ERASE GAP command.
                      0881  ; START_DSE - Prepare an MSCP packet to do an ERASE command.
                      0881  ;
                      0881  ; INPUTS:
                      0881  ;       R2 => MSCP buffer
                      0881  ;       R3 => UCB
                      0881  ;       R4 => PDT
                      0881  ;       R5 => CDRP
                      0881  ;
                      0881  ;       MSCP packet is zero except for MSCP$L_CMD_REF and MSCP$W_UNIT fields.
                      0881  ;
                      0881
                      0881 START_ERASETAPE:
              16  90  0881          MOVB    #MSCP$K_OP_ERGAP,-      ; Transfer ERASEGAP opcode
           08 A2      0883                  MSCP$B_OPCODE(R2)       ;  to packet.
              14  11  0885          BRB     WTM_ERASE_COM           ; Branch around to common.
                      0887
                      0887 START_DSE:
              12  90  0887          MOVB    #MSCP$K_OP_ERASE,-      ; Transfer ERASE opcode
           08 A2      0889                  MSCP$B_OPCODE(R2)       ;  to packet.
              07  E1  088B          BBC     #IO$V_NOWAIT,-          ; If NOT nowait, branch around.
           C0 A5      088D                  CDRP$W_FUNC(R5),-
              0B      088F                  WTM_ERASE_COM
                      0890          ASSUME  MSCP$V_MD_IMMED LE 7
 0A A2     40 8F  88  0890          BISB    #MSCP$M_MD_IMMED, -     ; If NOWAIT, then set proper TMSCP
                      0895                  MSCP$W_MODIFIER(R2)     ; modifier in command message.
              04  11  0895          BRB     WTM_ERASE_COM           ; Branch around to common.
                      0897
                      0897 START_WRITEMARK:
                      0897 START_WRITEOF:
              24  90  0897          MOVB    #MSCP$K_OP_WRITM,-      ; Transfer WRITE TAPE MARK opcode
           08 A2      0899                  MSCP$B_OPCODE(R2)       ;  to packet.
                      089B
                      089B WTM_ERASE_COM:
                      089B
                      089B          IF_IVCMD then=WRITM_IVCMD_END   ; Branch if invalid command processing.
                      089F
                      089F          SEND_MSCP_MSG                   ; Send message to remote MSCP server.
                      08A2
                      08A2          ASSUME  MT$V_BOT  GE 16
                      08A2          ASSUME  MT$V_EOF  GE 16
                      08A2          ASSUME  MT$V_EOT  GE 16
                      08A2          ASSUME  MT$V_LOST GE 16
   46 A3   17     8A  08A2          BICB    #<<MT$M_BOT ! MT$M_EOF -; Clear position sensitive DEVDEPEND
                      08A6                    ! MT$M_EOT -          ; bits
                      08A6                    ! MT$M_LOST> @ -16>, -
                      08A6                  UCB$L_DEVDEPEND+2(R3)
                      08A6
                      08A6          DO_ACTION       NONTRANSFER     ; Decode MSCP end status.
                      08A9          ACTION_ENTRY    SUCC,  SS$_NORMAL,      WRITM_SUCC
                      08AE          ACTION_ENTRY    ABRTD, SS$_ABORT,       WRITM_ABORT
                      08B3          ACTION_ENTRY    OFFLN, SS$_DEVOFFLINE,  WRITM_OFFLINE
                      08B8          ACTION_ENTRY    AVLBL, SS$_MEDOFL,      WRITM_AVAIL
                      08BD          ACTION_ENTRY    WRTPR, SS$_WRITLCK,     WRITM_WRITLCK
```

```
                       08C2              ACTION_ENTRY    PRESE, SS$_SERIOUSEXCP, WRITM_PRESE
                       08C7              ACTION_ENTRY    CNTLR, SS$_CTRLERR,     WRITM_CTRLERR
                       08CC              ACTION_ENTRY    FMTER, SS$_CTRLERR,     WRITM_FMTER
                       08D1              ACTION_ENTRY    DATA,  SS$_PARITY,      WRITM_DATA_ERROR
                       08D6              ACTION_ENTRY    DRIVE, SS$_DRVERR,      WRITM_DRVERR
                       08DB              ACTION_ENTRY    PLOST, SS$_CTRLERR,     ERASEGAP_PLOST
                       08E0              ACTION_ENTRY    ICMD,  SS$_CTRLERR,     WRITM_IVCMD
                       08E5              ACTION_ENTRY    END_TABLE
                       08E7
           078F   31   08E7              BRW     INVALID_STS              ; Unexpected MSCP end status.
                       08EA
                       08EA      WRITM_IVCMD:
                       08EA              IVCMD_BEGIN                      ; Begin invalid command processing.
           FD11   31   08ED              BRW     TU_BEGIN_IVCMD           ; Rebuild fatal MSCP command.
                       08F0      WRITM_IVCMD_END:
                       08F0              IVCMD_END                        ; Complete invalid command processing.
             14   11   08F2              BRB     WRITM_END                ; Branch around to end.
                       08F4
                       08F4      ERASEGAP_PLOST:
                       08F4              ASSUME  MT$V_LOST GE 16
      46 A3   10  88   08F4              BISB    #<MT$M_LOST @ -16>, -    ; Set position LOST DEVDEPEND bit.
                       08F8                      UCB$L_DEVDEPEND+2(R3)
                       08F8      WRITM_ABORT:
                       08F8      WRITM_OFFLINE:
                       08F8      WRITM_AVAIL:
                       08F8      WRITM_WRITLCK:
                       08F8      WRITM_CTRLERR:
                       08F8      WRITM_FMTER:
                       08F8      WRITM_DRVERR:
                       08F8      WRITM_DATA_ERROR:
                       08F8      WRITM_SUCC:
        00B0 C3   D5   08F8              TSTL    UCB$L_RECORD(R3)         ; Previously at BOT?
             04   12   08FC              BNEQ    10$                      ; Branch if not previously at BOT.
      40 A5   20  88   08FE              BISB    #CDRP$M_DENSCK, -        ; Else, set density check required flag.
                       0902                      CDRP$L_DUTUFLAGS(R5)
   00B0 C3   1C A2  D0 0902      10$:    MOVL    MSCP$L_POSITION(R2), -   ; Update tape position information.
                       0908                      UCB$L_RECORD(R3)
                       0908      WRITM_END:
             03   E1   0908              BBC     #MSCP$V_EF_EOT,-         ; See if we passed into End Of Tape
       0C 09 A2        090A                      MSCP$B_FLAGS(R2),40$     ;  region, and branch around if NOT.
                       090D              ASSUME  MT$V_EOT  GE 16
      46 A3   04  88   090D              BISB    #<MT$M_EOT @ -16>, -     ; Set EOT DEVDEPEND position bit.
                       0911                      UCB$L_DEVDEPEND+2(R3)
          05 50   E9   0911              BLBC    R0,40$                   ; If already an error, branch around.
   50   0878 8F   B0   0914              MOVW    #SS$_ENDOFTAPE,R0        ; Return EOT.
                       0919      40$:
                       0919      WRITM_PRESE:
           03AC   31   0919              BRW     FUNCTION_EXIT            ; Branch to common exit.
```

```
                        091C           .SBTTL  Start REWIND.
                        091C
                        091C    ; START_REWIND - Prepare an MSCP packet to do a REWIND command.
                        091C    ;
                        091C    ; A Rewind QIO request causes us to send an MSCP Reposition Command with
                        091C    ; the MSCP$M_MD_REWIND modifier set and both the MSCP$L_REC_CNT and
                        091C    ; MSCP$L_TMGP_CNT fields zero.  If the user specifies IO$M_NOWAIT, then
                        091C    ; the MSCPP$M_MD_IMMED modifier is set in the command that is sent.
                        091C    ;
                        091C    ; INPUTS:
                        091C    ;              R2 => MSCP buffer
                        091C    ;              R3 => UCB
                        091C    ;              R4 => PDT
                        091C    ;              R5 => CDRP
                        091C    ;
                        091C    ;              MSCP packet is zero except for MSCP$L_CMD_REF and MSCP$W_UNIT fields.
                        091C    ;
                        091C
                        091C    START_RECAL:
                        091C    START_REWIND:
                        091C
            25     90   091C           MOVB     #MSCP$K_OP_REPOS,-          ; Transfer REPOS.. ION opcode
         08 A2          091E                    MSCP$B_OPCODE(R2)           ;  to packet.
            02     A8   0920           BISW     #MSCP$M_MD_REWND,-          ; Specify rewind.
         0A A2          0922                    MSCP$W_MODIFIER(R2)
                        0924
            07     E1   0924           BBC      #IO$V_NOWAIT,-              ; If NOT nowait, branch around.
      05 C0 A5          0926                    CDRP$W_FUNC(R5),10$
                        0929           ASSUME   MSCP$V_MD_IMMED LE 7
0A A2    40 8F     88   0929           BISB     #MSCP$M_MD_IMMED, -         ; If NOWAIT, then set proper TMSCP
                        092E                    MSCP$W_MODIFIER(R2)         ; modifier in command message.
                        092E
                        092E    10$:   IF_IVCMD then=REWIND_IVCMD_END       ; Branch if invalid command processing.
                        0932
                        0932           SEND_MSCP_MSG                        ; Send message to remote MSCP server.
                        0935
                        0935           DO_ACTION        NONTRANSFER         ; Decode MSCP end status.
                        0938           ACTION_ENTRY     SUCC,  SS$_NORMAL,       REWIND_SUCC
                        093D           ACTION_ENTRY     ABRTD, SS$_ABORT,        REWIND_ABORT
                        0942           ACTION_ENTRY     PRESE, SS$_SERIOUSEXCP,  REWIND_PRESE
                        0947           ACTION_ENTRY     OFFLN, SS$_DEVOFFLINE,   REWIND_OFFLINE
                        094C           ACTION_ENTRY     AVLBL, SS$_MEDOFL,       REWIND_AVAIL
                        0951           ACTION_ENTRY     CNTLR, SS$_CTRLERR,      REWIND_CTRLERR
                        0956           ACTION_ENTRY     FMTER, SS$_CTRLERR,      REWIND_FMTER
                        095B           ACTION_ENTRY     DRIVE, SS$_DRVERR,       REWIND_DRVERR
                        0960           ACTION_ENTRY     ICMD,  SS$_CTRLERR,      REWIND_IVCMD
                        0965           ACTION_ENTRY     END_TABLE
                        0967
          070F     31   0967           BRW      INVALID_STS                 ; Unexpected MSCP end status.
                        096A
                        096A    REWIND_IVCMD:
                        096A           IVCMD_BEGIN                          ; Begin invalid command processing.
          FC91     31   096D           BRW      TU_BEGIN_IVCMD              ; Rebuild fatal MSCP command.
                        0970    REWIND_IVCMD_END:
                        0970           IVCMD_END                            ; Complete invalid command processing.
            10     11   0972           BRB      REWIND_END                  ; Branch around to end.
                        0974
```

```
                    0974  REWIND_SUCC:
       1C A2    D0  0974           MOVL    MSCP$L_POSITION(R2),-   ; Update positon on tape.
     00B0 C3        0977                   UCB$L_RECORD(R3)
          08    12  097A           BNEQ    30$                     ; This should be a NOP.
                    097C           ASSUME  MT$V_BOT  GE 16
                    097C           ASSUME  MT$V_EOF  GE 16
                    097C           ASSUME  MT$V_EOT  GE 16
                    097C           ASSUME  MT$V_LOST GE 16
  46 A3   16    8A  097C           BICB    #<<MT$M_EOF ! MT$M_EOT -; Clear position sensitive DEVDEPEND
                    0980                    ! MT$M_LOST> @ -16>, - ; bits.
                    0980                   UCB$L_DEVDEPEND+2(R3)
  46 A3   01    88  0980           BISB    #<MT$M_BOT @ -16>, -    ; Set BOT DEVDEPEND position bit.
                    0984                   UCB$L_DEVDEPEND+2(R3)
                    0984  30$:
                    0984  REWIND_ABORT:
                    0984  REWIND_OFFLINE:
                    0984  REWIND_AVAIL:
                    0984  REWIND_FMTER:
                    0984  REWIND_CTRLERR:
                    0984  REWIND_DRVERR:
                    0984  REWIND_PRESE:
                    0984  REWIND_END:
        0341    31  0984           BRW     FUNCTION_EXIT           ; Branch to common exit.
```

```
                      0987            .SBTTL  Start Space Records and Space Files.
                      0987
                      0987   ;+
                      0987   ; START_SPACEFILE   -
                      0987   ; START_SKIPFILE    - Prepare an MSCP packet to do a REPOSITION command
                      0987   ;                      so as to Skip files.
                      0987   ; START_SPACERECORD -
                      0987   ; START_SKIPRECORD  - Prepare an MSCP packet to do a REPOSITION command
                      0987   ;                      so as to Skip records.
                      0987   ;
                      0987   ; INPUTS:
                      0987   ;         R2 => MSCP buffer
                      0987   ;         R3 => UCB
                      0987   ;         R4 => PDT
                      0987   ;         R5 => CDRP
                      0987   ;         CDRP$L_MEDIA = # of records or files to
                      0987   ;                         skip (word count in longword field).
                      0987   ;
                      0987   ;         MSCP packet is zero except for MSCP$L_CMD_REF and MSCP$W_UNIT fields.
                      0987   ;-
                      0987
                      0987  START_SKIPFILE:
                      0987  START_SPACEFILE:
                      0987
   51    10 A2    9E  0987          MOVAB   MSCP$L_TMGP_CNT(R2),R1   ; R1 => field to fill in for skip files.
            04    11  098B          BRB     SKIP_COMMON              ; Branch around to common code.
                      098D
                      098D  START_SKIPRECORD:
                      098D  START_SPACERECORD:
                      098D
   51    0C A2    9E  098D          MOVAB   MSCP$L_REC_CNT(R2),R1    ; R1 => field to fill in for skip records.
                      0991
                      0991  SKIP_COMMON:
            25    90  0991          MOVB    #MSCP$K_OP_REPOS,-       ; Transfer REPOSITION opcode
         08 A2        0993                  MSCP$B_OPCODE(R2)        ;  to packet.
   50    D8 A5    32  0995          CVTWL   CDRP$L_MEDIA(R5),R0      ; Pickup # records to skip.
            09    18  0999          BGEQ    10$                      ; GEQ implies positive (forward) movement.
      50    50    CE  099B          MNEGL   R0,R0                    ; Get absolute value of # to skip.
            08    A8  099E          BISW    #MSCP$M_MD_REVRS,-       ; Set modifier to indicate reverse
         0A A2        09A0                  MSCP$W_MODIFIER(R2)      ;  motion.
            14    11  09A2          BRB     17$                      ; If reverse, then do NOT try to detect
                      09A4                                           ;  LEOT, so branch around.
                      09A4
                      09A4  10$:    ; Detect LEOT is performed on all tapes NOT mounted ANSI.  That is,
                      09A4          ; all tapes either NOT mounted or mounted Foreign.  The only exception
                      09A4          ; is for physical I/O requests.
                      09A4
0F CA A5    08    E0  09A4          BBS     #IRP$V_PHYSIO, -         ; If physical I/O function, branch
                      09A9                  CDRP$W_STS(R5), 17$      ;  around setting to Detect LEOT.
05 38 A3    13    E1  09A9          BBC     #DEV$V_MNT, -            ; If Tape NOT mounted, go try to Detect
                      09AE                  UCB$L_DEVCHAR(R3), 14$   ; LEOT.
05 38 A3    18    E1  09AE          BBC     #DEV$V_FOR, -            ; If NOT foreign, than ANSI, so branch
                      09B3                  UCB$L_DEVCHAR(R3), 17$   ; around setting to Detect LEOT.
                      09B3  14$:    ASSUME  MSCP$V_MD_DLEOT LE 7
0A A2    80 8F    88  09B3          BISB    #MSCP$M_MD_DLEOT, -      ; Set modifier to ask to Detect LEOT.
                      09B8                  MSCP$W_MODIFIER(R2)
                      09B8
      61   50   D0  09B8  17$:    MOVL    R0, (R1)                      ; Put #records(files) to skip in packet.
                    09BB
                    09BB          IF_IVCMD then=SKIP_IVCMD_END          ; Branch if invalid command processing.
                    09BF
                    09BF          SEND_MSCP_MSG                         ; Send message to remote MSCP server.
                    09C2
                    09C2          ASSUME  MT$V_BOT  GE 16
                    09C2          ASSUME  MT$V_EOF  GE 16
                    09C2          ASSUME  MT$V_EOT  GE 16
                    09C2          ASSUME  MT$V_LOST GE 16
   46 A3   17   8A  09C2          BICB    #<<MT$M_BOT ! MT$M_EOF -      ; Clear position sensitive DEVDEPEND
                    09C6             ! MT$M_EOT -                       ; bits
                    09C6             ! MT$M_LOST> @ -16>, -
                    09C6            UCB$L_DEVDEPEND+2(R3)
                    09C6
                    09C6          DO_ACTION       TRANSFER              ; Decode MSCP end status.
                    09C9          ACTION_ENTRY    SUCC,  SS$_NORMAL,        SKIP_SUCC
                    09CE          ACTION_ENTRY    LED,   SS$_ENDOFVOLUME,   SKIP_LEOT
                    09D3          ACTION_ENTRY    ABRTD, SS$_ABORT,         SKIP_ABORT
                    09D8          ACTION_ENTRY    PRESE, SS$_SERIOUSEXCP,   SKIP_PRESE
                    09DD          ACTION_ENTRY    OFFLN, SS$_DEVOFFLINE,    SKIP_OFFLINE
                    09E2          ACTION_ENTRY    AVLBL, SS$_MEDOFL,        SKIP_AVAIL
                    09E7          ACTION_ENTRY    CNTLR, SS$_CTRLERR,       SKIP_CTRLERR
                    09EC          ACTION_ENTRY    FMTER, SS$_CTRLERR,       SKIP_FMTER
                    09F1          ACTION_ENTRY    DRIVE, SS$_DRVERR,        SKIP_DRVERR
                    09F6          ACTION_ENTRY    BOT,   SS$_NORMAL,        SKIP_BOT
                    09FB          ACTION_ENTRY    TAPEM, SS$_ENDOFFILE,     SKIP_EOF
                    0A00          ACTION_ENTRY    PLOST, SS$_CTRLERR,       SKIP_PLOST
                    0A05          ACTION_ENTRY    ICMD,  SS$_CTRLERR,       SKIP_IVCMD
                    0A0A          ACTION_ENTRY    END_TABLE
                    0A0C
         066A   31  0A0C          BRW     INVALID_STS                   ; Unexpected MSCP end status.
                    0A0F
                    0A0F  SKIP_IVCMD:
                    0A0F          IVCMD_BEGIN                           ; Begin invalid command processing.
         FBEC   31  0A12          BRW     TU_BEGIN_IVCMD                ; Rebuild fatal MSCP command.
                    0A15  SKIP_IVCMD_END:
                    0A15          IVCMD_END                             ; Complete invalid command processing.
                    0A17  ; ----- BRB     SKIP_ABORT                    ; Fall through to finish skip operation.
                    0A17  SKIP_PRESE:
                    0A17  SKIP_ABORT:
                    0A17  SKIP_OFFLINE:
                    0A17  SKIP_AVAIL:
 50   50   10   9C  0A17          ROTL    #16,R0,R0                     ; Move SS$_ code into low order.
           34   11  0A1B          BRB     SKIP_END                      ; Branch around to end.
                    0A1D
                    0A1D  SKIP_PLOST:
                    0A1D          ASSUME  MT$V_LOST GE 16
   46 A3   10   88  0A1D          BISB    #<MT$M_LOST @ -16>, -         ; Set position LOST DEVDEPEND bit.
                    0A21            UCB$L_DEVDEPEND+2(R3)
           0A   11  0A21          BRB     SKIP_SUCC                     ; Rejoin common code.
                    0A23  SKIP_EOF:
                    0A23          ASSUME  MT$V_EOF  GE 16
   46 A3   02   88  0A23          BISB    #<MT$M_EOF @ -16>, -          ; Set EOF DEVDEPEND position bit.
                    0A27            UCB$L_DEVDEPEND+2(R3)
           04   11  0A27          BRB     SKIP_SUCC                     ; Rejoin common code.
                    0A29  SKIP_BOT:
                         0A29                 ASSUME  MT$V_BOT  GE 16
       46 A3    01    88 0A29                 BISB    #<MT$M_BOT @ -16>, -       ; Set BOT DEVDEPEND position bit.
                         0A2D                         UCB$L_DEVDEPEND+2(R3)
                         0A2D       ; ----- BRB       SKIP_SUCC                  ; Rejoin common code.
                         0A2D       SKIP_FMTER:
                         0A2D       SKIP_CTRLERR:
                         0A2D       SKIP_DRVERR:
                         0A2D       SKIP_SUCC:
                         0A2D       SKIP_LEOT:
    04 09 A2    03    E1 0A2D                 BBC     #MSCP$V_EF_EOT, -          ; Is tape in the EOT region?
                         0A32                         MSCP$B_FLAGS(R2), 10$      ; Branch if tape not in EOT.
                         0A32                 ASSUME  MT$V_EOT  GE 16
       46 A3    04    88 0A32                 BISB    #<MT$M_EOT @ -16>, -       ; Else, set EOT DEVDEPEND position bit.
                         0A36                         UCB$L_DEVDEPEND+2(R3)
                         0A36
          00B0 C3     D5 0A36       10$:      TSTL    UCB$L_RECORD(R3)           ; Previously at BOT?
                  04  12 0A3A                 BNEQ    15$                        ; Branch if not previously at BOT.
       40 A5    20    88 0A3C                 BISB    #CDRP$M_DENSCK, -          ; Else, set density check required flag.
                         0A40                         CDRP$L_DUTUFLAGS(R5)
  00B0 C3    1C A2    D0 0A40       15$:      MOVL    MSCP$L_POSITION(R2), -     ; Update tape position information.
                         0A46                         UCB$L_RECORD(R3)
             0C A2    C1 0A46                 ADDL3   MSCP$L_RCSKIPED(R2),-      ; Add records and tapemarks skipped
       51    10 A2       0A49                         MSCP$L_TMSKIPED(R2),R1     ;  so as to return to user.
50     50    F0 8F    79 0A4C                 ASHQ    #-16,R0,R0                 ; Shift count and SS$_ code into position.
                         0A51       SKIP_END:
             0274     31 0A51                 BRW     FUNCTION_EXIT              ; Branch to common exit.
```

```
                          0A54          .SBTTL  Start a SETCHAR or a SETMODE function
                          0A54
                          0A54  ; START_SETCHAR and START_SETMODE
                          0A54  ;       The quad-word of data for the operation is contained in IRP$L_MEDIA.
                          0A54  ;       This "PHYSICAL" I/O function and the "LOGICAL" I/O function
                          0A54  ;       SET MODE are almost identical.  The only difference is that while
                          0A54  ;       both allow for the setting of:
                          0A54  ;
                          0A54  ;               1. Default buffer size
                          0A54  ;               2. Tape density (1600 BPI or 6250 BPI).
                          0A54  ;               3. Tape format
                          0A54  ;               4. Serious Exception mode
                          0A54  ;
                          0A54  ;       the former function (i.e. SET CHARACTERISTICS) also allows for
                          0A54  ;       the resetting of the DEVICE CLASS and the DEVICE TYPE fields in
                          0A54  ;       the UCB.
                          0A54  ;
                          0A54  ;       The first two bytes of the QUADWORD of data at IRP$L_MEDIA contain
                          0A54  ;       the DEVICE CLASS and DEVICE TYPE respectively for a SETCHAR.
                          0A54  ;       The next word of the QUADWORD contains the new buffer size.  The
                          0A54  ;       third word contains new density and format information.  The fourth
                          0A54  ;       word of the QUADWORD is reserved.
                          0A54  ;
                          0A54  ; INPUTS:
                          0A54  ;       R2 => MSCP buffer
                          0A54  ;       R3 => UCB
                          0A54  ;       R4 => PDT
                          0A54  ;       R5 => CDRP
                          0A54  ;
                          0A54
                          0A54  START_SETCHAR:
                          0A54          ASSUME  UCB$B_DEVTYPE EQ UCB$B_DEVCLASS+1
  40 A3    D8 A5    B0    0A54          MOVW    CDRP$L_MEDIA(R5),UCB$B_DEVCLASS(R3)      ; Reset CLASS and TYPE.
                          0A59
                          0A59  START_SETMODE:
  42 A3    DA A5    B0    0A59          MOVW    CDRP$L_MEDIA+2(R5),UCB$W_DEVBUFSIZ(R3)  ; Copy new buffer size.
                          0A5E
                          0A5E          START_SEQNOP                            ; Synchronize class driver - server
                          0A74                                                  ; communications so that only this
                          0A74                                                  ; thread is sending commands to the
                          0A74                                                  ; server.
                          0A74          ASSUME  CDRP$V_CAND EQ 0
       22 40 A5     E8    0A74          BLBS    CDRP$L_DUTUFLAGS(R5), - ; Was I/O request canceled?
                          0A78                  SETMODE_CANCEL          ; Branch if request was canceled.
             03     90    0A78          MOVB    #MSCP$K_OP_GTUNT,-      ; Opcode is for GET UNIT STATUS.
          08 A2           0A7A                  MSCP$B_OPCODE(R2)
                          0A7C          ASSUME  MSCP$V_MD_CLSEX GE 8
             20     8A    0A7C          BICB    #<MSCP$M_MD_CLSEX@-8>,- ; The clear serious execption modifier
          08 A2           0A7E                  MSCP$W_MODIFIER+1(R2)   ; is illegal on get unit status cmds.
                          0A80          SEND_MSCP_MSG                   ; Send message to remote MSCP server.
                          0A83
                          0A83          IF_MSCP SUCCESS, then=SETMODE_ONLINE  ; Branch if GTUNT successful.
                          0A89          .IF     DF      TU_SEQCHK       ; Override sequence checking and
                          0A89          BSBW    OVERRIDE_SEQCHK         ; remove sequence number from array.
                          0A89          .ENDC
  50    01A4 8F     3C    0A89          MOVZWL  #SS$_MEDOFL, R0         ; Setup final I/O status.
                          0A8E
```

```
                 0A8E  SETMODE_ABORT:
                 0A8E  SETMODE_OFFLINE:
                 0A8E  SETMODE_CTRLERR:
                 0A8E  SETMODE_DRVERR:
          08  EF 0A8E          EXTZV   #MT$V_DENSITY,-
          05     0A90                  #MT$S_DENSITY,-
 51    DC A5     0A91                  CDRP$L_MEDIA+4(R5),R1   ; Extract user designated DENSITY parameter.
          51  F0 0A94          INSV    R1,-                    ; And insure that UCB$L_DEVDEPEND winds
                 0A96                  #MT$V_DENSITY,-         ;  up with the correct value for DENSITY
     05   08     0A96                  #MT$S_DENSITY,-
       44 A3     0A98                  UCB$L_DEVDEPEND(R3)
                 0A9A
                 0A9A  SETMODE_CANCEL:
        00B0  31 0A9A          BRW     SETMODE_RETURN          ; And branch around.
                 0A9D
                 0A9D  SETMODE_ONLINE:
                 0A9D
                 0A9D          ASSUME  CDRP$V_CAND EQ 0
    ED 40 A5  E8 0A9D          BLBS    CDRP$L_DUTUFLAGS(R5), - ; Was I/O request canceled?
                 0AA1                  SETMODE_ABORT           ; Branch if request was canceled.
          02  E0 0AA1          BBS     #MT$V_ENSEREXCP,-       ; Branch if Serious Exception explicitly
    06 DC A5     0AA3                  CDRP$L_MEDIA+4(R5),10$  ;  enabled.
          04  CA 0AA6          BICL    #MT$M_ENSEREXCP,-       ; Else clear Serious Exception mode.
       44 A3     0AA8                  UCB$L_DEVDEPEND(R3)
          04  11 0AAA          BRB     20$                     ; And branch around.
                 0AAC  10$:
          04  C8 0AAC          BISL    #MT$M_ENSEREXCP,-       ; Enable Serious Exception mode.
       44 A3     0AAE                  UCB$L_DEVDEPEND(R3)
                 0AB0  20$:
       20 A2  B0 0AB0          MOVW    MSCP$W_FORMAT(R2),-     ; Copy format to UCB before recycling
     00F0 C3     0AB3                  UCB$W_TU_FORMAT(R3)     ;  end message.
                 0AB6
                 0AB6          RESET_MSCP_MSG                  ; Setup message buf. etc. for reuse.
                 0AB9
                 0AB9  SETMODE_BEGIN_IVCMD:
                 0AB9
          0A  90 0AB9          MOVB    #MSCP$K_OP_STUNT,-      ; Transfer Set Unit Characteristics
       08 A2     0ABB                  MSCP$B_OPCODE(R2)       ;  opcode to packet.
                 0ABD
     00E0 C3  B0 0ABD          MOVW    UCB$W_UNIT_FLAGS(R3),-  ; Copy unit flags to MSCP packet.
       0E A2     0AC1                  MSCP$Q_UNT_FLGS(R2)
                 0AC3
     00D8 C3  D0 0AC3          MOVL    UCB$L_MSCPDEVPARAM(R3),-; Copy Device dependent parameters to
       1C A2     0AC7                  MSCP$L_DEV_PARM(R2)     ;  MSCP packet.
                 0AC9
     00B0 C3  D5 0AC9          TSTL    UCB$L_RECORD(R3)        ; Is tape at BOT?
          19  12 0ACD          BNEQ    35$                     ; Skip density setup if not at BOT.
          08  EF 0ACF          EXTZV   #MT$V_DENSITY,-         ; Determine density that the user has
          05     0AD1                  #MT$S_DENSITY,-         ;  specified for this unit
 50    DC A5     0AD2                  CDRP$L_MEDIA+4(R5),R0   ;  and put into R0.
                 0AD5
        F934  30 0AD5          BSBW    VMSTOMSCP_DENS          ; Convert VMS density to MSCP format.
       09 50  E8 0AD8          BLBS    R0,30$                  ; LBS means successful conversion.
          08  EF 0ADB          EXTZV   #MT$V_DENSITY,-         ; Determine density that the user has
          05     0ADD                  #MT$S_DENSITY,-         ;  last established for this unit
 50    44 A3     0ADE                  UCB$L_DEVDEPEND(R3),R0  ;  and put into R0.
        F928  30 0AE1          BSBW    VMSTOMSCP_DENS          ; Convert VMS density to MSCP format.
```

```
                           0AE4      30$:
        20 A2    51    B0  0AE4                 MOVW    R1,MSCP$W_FORMAT(R2)     ; Copy MSCP density to packet.
                           0AE8
                           0AE8      35$:       ASSUME  MT$K_SPEED_DEF EQ 0
                 18    EF  0AE8                 EXTZV   #MT$V_SPEED,-            ; Extract user specified speed.
                 08        0AEA                         #MT$S_SPEED,-
        50    DC A5        0AEB                         CDRP$L_MEDIA+4(R5),R0
                 09    13  0AEE                 BEQL    40$                      ; EQL implies default.
               F93D    30  0AF0                 BSBW    SPEEDTOMSCP              ; Convert speed to MSCP format.
                 20    A8  0AF3                 BISW    #MSCP$M_UF_VSMSU,-       ; Enable variable speed mode suppression.
              0E A2        0AF5                         MSCP$W_UNT_FLGS(R2)
                 04    11  0AF7                 BRB     50$                      ; And branch around.
                           0AF9      40$:
                 20    AA  0AF9                 BICW    #MSCP$M_UF_VSMSU,-       ; Disable variable speed mode suppression.
              0E A2        0AFB                         MSCP$W_UNT_FLGS(R2)
                           0AFD      50$:
        22 A2    50    B0  0AFD                 MOVW    R0,MSCP$W_SPEED(R2)      ; Place speed value into packet.
                           0B01
               F966    30  0B01                 BSBW    SET_CLEAR_SEX            ; Set SEX if called for.
                           0B04
                           0B04                 IF_IVCMD then=SETMODE_IVCMD_END ; Branch if invalid command processing.
                           0B08
                           0B08                 SEND_MSCP_MSG                    ; Send message to remote MSCP server.
                           0B0B
                           0B0B                 DO_ACTION       NONTRANSFER     ; Decode MSCP end status.
                           0B0E                 ACTION_ENTRY    SUCC,  SS$_NORMAL,      SETMODE_SUCC
                           0B13                 ACTION_ENTRY    PRESE, SS$_SERIOUSEXCP, SETMODE_RETURN
                           0B18                 ACTION_ENTRY    ABRTD, SS$_ABORT,       SETMODE_ABORT
                           0B1D                 ACTION_ENTRY    ICMD,  SS$_BUGCHECK,    SETMODE_IVCMD
                           0B22                 ACTION_ENTRY    OFFLN, SS$_MEDOFL,      SETMODE_OFFLINE
                           0B27                 ACTION_ENTRY    AVLBL, SS$_MEDOFL,      SETMODE_OFFLINE
                           0B2C                 ACTION_ENTRY    CNTLR, SS$_CTRLERR,     SETMODE_CTRLERR
                           0B31                 ACTION_ENTRY    FMTER, SS$_CTRLERR,     SETMODE_CTRLERR
                           0B36                 ACTION_ENTRY    DRIVE, SS$_DRVERR,      SETMODE_DRVERR
                           0B3B                 ACTION_ENTRY    END_TABLE
                           0B3D
               0539    31  0B3D                 BRW     INVALID_STS              ; Unexpected MSCP end status.
                           0B40
                           0B40      SETMODE_IVCMD:
                           0B40                 IVCMD_BEGIN                      ; Begin invalid command processing.
               FF73    31  0B43                 BRW     SETMODE_BEGIN_IVCMD      ; Rebuild fatal MSCP command.
                           0B46      SETMODE_IVCMD_END:
                           0B46                 IVCMD_END                        ; Complete invalid command processing.
                 03    11  0B48                 BRB     SETMODE_RETURN           ; Complete setmode operation.
                           0B4A
                           0B4A      SETMODE_SUCC:
                           0B4A
               FC48    30  0B4A                 BSBW    RECORD_SETUNIT_CHAR      ; Record data from End Message in UCB.
                           0B4D
                           0B4D      SETMODE_RETURN:
                           0B4D                 END_SEQNOP                       ; End synchronized class driver -
                           0B63                                                  ; server communications.
               0162    31  0B63                 BRW     FUNCTION_EXIT            ; Terminate I/O request.
```

```
                        0B66          .SBTTL  Start SENSECHAR and SENSEMODE functions.
                        0B66
                        0B66  ; START_SENSECHAR and START_SENSEMODE.
                        0B66  ;
                        0B66  ; INPUTS:
                        0B66  ;       R2 => MSCP buffer
                        0B66  ;       R3 => UCB
                        0B66  ;       R4 => PDT
                        0B66  ;       R5 => CDRP
                        0B66  ;
                        0B66
                        0B66 START_SENSECHAR:
                        0B66 START_SENSEMODE:
                        0B66
          03     90     0B66          MOVB    #MSCP$K_OP_GTUNT,-       ; Opcode is for GET UNIT STATUS.
       08 A2            0B68                  MSCP$B_OPCODE(R2)
                        0B6A          ASSUME  MSCP$V_MD_CLSEX GE 8
          20     8A     0B6A          BICB    #<MSCP$M_MD_CLSEX@-8>,- ; The clear serious execption modifier
       08 A2            0B6C                  MSCP$W_MODIFIER+1(R2)   ; is illegal on get unit status cmds.
                        0B6E          SEND_MSCP_MSG                    ; Send message to remote MSCP server.
                        0B71
                        0B71          IF_MSCP SUCCESS, then=SENSEMODE_ONLINE  ; Branch if GTUNT successful.
 50   01A4 8F    3C     0B77          MOVZWL  #SS$_MEDOFL,R0           ; Mark final I/O status.
          06     11     0B7C          BRB     SENSEMODE_RETURN         ; And branch around.
                        0B7E
                        0B7E SENSEMODE_ONLINE:
                        0B7E
        FC22     30     0B7E          BSBW    RECORD_GETUNIT_CHAR      ; Copy data from End Message to UCB.
    50    01     3C     0B81          MOVZWL  #SS$_NORMAL, R0          ; Setup successful completion status.
                        0B84
                        0B84 SENSEMODE_RETURN:
        0141     31     0B84          BRW     FUNCTION_EXIT
```

```
                       0B87             .SBTTL  START_READPBLK and START_WRITEPBLK and START_WRITECHECK
                       0B87
                       0B87      ; START_READPBLK - Prepare an MSCP packet to do a READ command.
                       0B87      ;
                       0B87      ; START_WRITEPBLK - Prepare an MSCP packet to do a WRITE command.
                       0B87      ;
                       0B87      ; START_WRITECHECK - Prepare an MSCP packet to do a COMPARE HOST DATA command.
                       0B87      ;
                       0B87      ; INPUTS:
                       0B87      ;       R2 => MSCP buffer
                       0B87      ;       R3 => UCB
                       0B87      ;       R4 => PDT
                       0B87      ;       R5 => CDRP
                       0B87      ;
                       0B87      ;       MSCP packet is zero except for MSCP$L_CMD_REF and MSCP$W_UNIT fields.
                       0B87      ;
                       0B87
                       0B87             .enable lsb
                       0B87      START_WRITECHECK:
                       0B87
            20    90   0B87             MOVB    #MSCP$K_OP_COMP,-       ; Compare host data opcode
         08 A2         0B89                     MSCP$B_OPCODE(R2)       ;  to packet.
            06    E1   0B8B             BBC     #IO$V_REVERSE,-         ; Branch around if NOT reverse.
      23 C0 A5         0B8D                     CDRP$Q_FUNC(R5),20$
            08    A8   0B90             BISW    #MSCP$M_MD_REVRS,-      ; Else set reverse modifier.
         0A A2         0B92                     MSCP$W_MODIFIER(R2)
            1D    11   0B94             BRB     20$                     ; And branch around to join common code
                       0B96
                       0B96      START_WRITEPBLK:
                       0B96
            22    90   0B96             MOVB    #MSCP$K_OP_WRITE,-      ; Transfer WRITE opcode
         08 A2         0B98                     MSCP$B_OPCODE(R2)       ;  to packet.
            0D    11   0B9A             BRB     10$
                       0B9C
                       0B9C      START_READPBLK:
                       0B9C
            21    90   0B9C             MOVB    #MSCP$K_OP_READ,-       ; Transfer READ opcode
         08 A2         0B9E                     MSCP$B_OPCODE(R2)       ;  to packet.
                       0BA0
            06    E1   0BA0             BBC     #IO$V_REVERSE,-         ; Branch around if NOT reverse.
      04 C0 A5         0BA2                     CDRP$Q_FUNC(R5),10$
            08    A8   0BA5             BISW    #MSCP$M_MD_REVRS,-      ; Else set reverse modifier.
         0A A2         0BA7                     MSCP$W_MODIFIER(R2)
                       0BA9      10$:
                       0BA9
            0E    E1   0BA9             BBC     #IO$V_DATACHECK,-       ; See if user specified compare in
      05 C0 A5         0BAB                     CDRP$Q_FUNC(R5),20$     ;  addition to data transfer. If not, branch
                       0BAE             ASSUME  MSCP$V_MD_COMP GE 8     ; Else, set the read/write with
0B A2   40 8F     88   0BAE             BISB    #<MSCP$M_MD_COMP@-8>, - ; data compare modifier.
                       0BB3                     MSCP$W_MODIFIER+1(R2)
                       0BB3      20$:
                       0BB3             IF_IVCMD then=70$               ; Branch if invalid command processing.
                       0BB7
      30 A5       9E   0BB7             MOVAB   CDRP$T_LBUFHNDL(R5),-   ; Put address of Local BUFfer HaNDLe
      2C A5            0BBA                     CDRP$L_LBUFH_AD(R5)     ;  field into field that points to it.
                       0BBC             MAP_IRP                         ; Allocate mapping resources and load
                       0BBF                                             ;  them with data from SVAPTE, BOFF.
```

```
                  0BBF                                              ;  and BCNT derived from IRP within
                  0BBF                                              ;  CDRP.
                  0BBF
52    1C A5   D0  0BBF          MOVL    CDRP$L_MSG_BUF(R5),R2       ; Refresh R2 => MSCP packet.
      30 A5   7D  0BC3   70$:   MOVQ    CDRP$T_LBUFHNDL(R5),-       ; Copy contents of buffer handle to
      10 A2       0BC6                  MSCP$B_BUFFER(R2)           ;  MSCP buffer descriptor field.
      38 A5   D0  0BC8          MOVL    CDRP$T_LBUFHNDL+8(R5),-     ; Buffer handle is 96 bits (12 bytes)
      18 A2       0BCB                  MSCP$B_BUFFER+8(R2)         ;  in length.
      D2 A5   D0  0BCD          MOVL    CDRP$L_BCNT(R5),-
      0C A2       0BD0                  MSCP$L_BYTE_CNT(R2)         ; Copy byte count of transfer.
                  0BD2
                  0BD2          IF_IVCMD then=XFER_IVCMD_END        ; Branch if invalid command processing.
                  0BD6
                  0BD6          .enable lsb                         ; Start a new local symbol block.
                  0BD6
                  0BD6          SEND_MSCP_MSG                       ; Send message to remote MSCP server.
                  0BD9
                  0BD9          ASSUME  MT$V_BOT  GE 16
                  0BD9          ASSUME  MT$V_EOF  GE 16
                  0BD9          ASSUME  MT$V_EOT  GE 16
                  0BD9          ASSUME  MT$V_LOST GE 16
46 A3   17    8A  0BD9          BICB    #<<MT$M_BOT ! MT$M_EOF -    ; Clear position sensitive DEVDEPEND
                  0BDD                    ! MT$M_EOT -              ; bits.
                  0BDD                    ! MT$M_LOST> @ -16>, -
                  0BDD                  UCB$L_DEVDEPEND+2(R3)
                  0BDD
                  0BDD          DO_ACTION       TRANSFER            ; Decode MSCP end status.
                  0BE0          ACTION_ENTRY    SUCC,  SS$_NORMAL,       TRANSFER_RTN_RECLEN
                  0BE5          ACTION_ENTRY    PRESE, SS$_SERIOUSEXCP,  TRANSFER_PRESE
                  0BEA          ACTION_ENTRY    ABRTD, SS$_ABORT,        TRANSFER_RTN_BCNT
                  0BEF          ACTION_ENTRY    ICMD,  SS$_CTRLERR,      TRANSFER_INVALID_COMMAND
                  0BF4          ACTION_ENTRY    COMP,  SS$_DATACHECK,    TRANSFER_COMPERR
                  0BF9          ACTION_ENTRY    OFFLN, SS$_MEDOFL,       TRANSFER_MEDOFL
                  0BFE          ACTION_ENTRY    AVLBL, SS$_MEDOFL,       TRANSFER_MEDOFL
                  0C03          ACTION_ENTRY    TAPEM, SS$_ENDOFFILE,    TRANSFER_EOF
                  0C08          ACTION_ENTRY    BOT,   SS$_ENDOFFILE,    TRANSFER_BOT
                  0C0D          ACTION_ENTRY    PLOST, SS$_CTRLERR,      TRANSFER_PLOST
                  0C12          ACTION_ENTRY    RDTRN, SS$_DATAOVERUN,   TRANSFER_RTN_RECLEN
                  0C17          ACTION_ENTRY    DATA,  SS$_PARITY,       TRANSFER_DATA_ERROR
                  0C1C          ACTION_ENTRY    HSTBF, SS$_IVBUFLEN,     TRANSFER_HOST_BUFFER_ERROR
                  0C21          ACTION_ENTRY    CNTLR, SS$_CTRLERR,      TRANSFER_CTRLERR
                  0C26          ACTION_ENTRY    FMTER, SS$_CTRLERR,      TRANSFER_RTN_BCNT
                  0C2B          ACTION_ENTRY    DRIVE, SS$_DRVERR,       TRANSFER_RTN_BCNT
                  0C30          ACTION_ENTRY    WRTPR, SS$_WRITLCK,      TRANSFER_RTN_BCNT
                  0C35          ACTION_ENTRY    END_TABLE
                  0C37
       043F   31  0C37          BRW     INVALID_STS                 ; Unexpected MSCP end status.
                  0C3A
                  0C3A   XFER_IVCMD_END:
         3A   11  0C3A          BRB     TRANSFER_IVCMD_END          ; Branch assist.
                  0C3C
                  0C3C
                  0C3C   TRANSFER_PLOST:
                  0C3C          ASSUME  MT$V_LOST GE 16
46 A3   10    88  0C3C          BISB    #<MT$M_LOST @ -16>, -       ; Set position LOST DEVDEPEND bit.
                  0C40                  UCB$L_DEVDEPEND+2(R3)
         0A   11  0C40          BRB     300$                        ; Join common code.
```

```
                            0C42  TRANSFER_EOF:
                            0C42          ASSUME  MT$V_EOF  GE 16
         46 A3    02    88  0C42          BISB    #<MT$M_EOF @ -16>, -     ; Set EOF DEVDEPEND position bit.
                            0C46                  UCB$L_DEVDEPEND+2(R3)
                  04    11  0C46          BRB     300$                     ; Join common code.
                            0C48  TRANSFER_BOT:
                            0C48          ASSUME  MT$V_BOT  GE 16
         46 A3    01    88  0C48          BISB    #<MT$M_BOT @ -16>, -     ; Set BOT DEVDEPEND position bit.
                            0C4C                  UCB$L_DEVDEPEND+2(R3)
                            0C4C  ; ----- BRB     300$                     ; Join common code.
                            0C4C
                  51    D4  0C4C  300$:   CLRL    R1                       ; Set zero bytes transfered.
                0049    31  0C4E          BRW     TRANSFER_SHIFT           ; Branch around.
                            0C51
                            0C51  TRANSFER_PRESE:
                            0C51
                  51    D4  0C51          CLRL    R1                       ; R1 = number of bytes transferred.
   50     50   F0 8F    79  0C53          ASHQ    #-16,R0,R0               ; Shift into proper position for IOSB.
                006D    31  0C58          BRW     FUNCTION_EXIT            ; Complete function immedately.
                            0C5B
                            0C5B  TRANSFER_CTRLERR:
                  05    EF  0C5B          EXTZV   #MSCP$S_ST_MASK,-        ; Extract the sub-code only.
                  0B        0C5D                  #16-MSCP$S_ST_MASK,-
          51    0A A2       0C5E                  MSCP$W_STATUS(R2),R1
            51    01    B1  0C61          CMPW    #MSCP$K_SC_DLATE,R1      ; Compare to Data Late error.
                  07    12  0C64          BNEQ    25$                      ; Branch around if not Data Late.
   50    22740000 8F    D0  0C66          MOVL    #SS$_DATALATE@16,R0      ; Set SS$_DATALATE into high word.
                002A    31  0C6D  25$:    BRW     TRANSFER_SHIFT           ; Branch to common code.
                            0C70
                            0C70  TRANSFER_INVALID_COMMAND:
                            0C70
                            0C70          IVCMD_BEGIN                      ; Begin invalid command processing.
                F9BB    31  0C73          BRW     TU_BEGIN_IVCMD           ; Rebuild fatal MSCP command.
                            0C76  TRANSFER_IVCMD_END:
                            0C76          IVCMD_END                        ; Complete invalid command processing.
                  D2    11  0C78          BRB     300$                     ; Complete the function.
                            0C7A
                            0C7A  TRANSFER_MEDOFL:
                            0C7A
                  06    E1  0C7A          BBC     #MSCP$V_SC_INOPR,-       ; Branch around if NOT unit inoperative
                0A A2       0C7C                  MSCP$W_STATUS(R2),-      ;  substatus.
                  17        0C7E                  TRANSFER_RTN_BCNT
   50    008C0000 8F    D0  0C7F          MOVL    #SS$_DRVERR@16,R0        ; Else set up R0 with proper SS$_ code
                            0C86                                           ;  in high order word and
                  0E    11  0C86          BRB     TRANSFER_RTN_BCNT        ; Branch around.
                            0C88  TRANSFER_HOST_BUFFER_ERROR:
                            0C88
                  05    EF  0C88          EXTZV   #MSCP$S_ST_MASK,-        ; Extract the sub-code only.
                  0B        0C8A                  #16-MSCP$S_ST_MASK,-
          51    0A A2       0C8B                  MSCP$W_STATUS(R2),R1
            51    02    B1  0C8E          CMPW    #MSCP$K_SC_ODDBC,R1      ; Compare to Odd Byte Count error.
                  03    13  0C91          BEQL    TRANSFER_RTN_BCNT        ; Branch around if Odd BCNT.
                03E3    31  0C93          BRW     INVALID_STS              ; Here we got an invalid MSCP status.
                            0C96
                            0C96  TRANSFER_DATA_ERROR:                     ; TRANSFER action routine for MSCP$K_ST_DATA
                            0C96
                            0C96  TRANSFER_COMPERR:
```

```
                         0C96  3138 TRANSFER_RTN_BCNT:
                         0C96  3139 TRANSFER_RTN_RECLEN:                                   ; Common TRANSFER action routine.
                         0C96  3140                                                        ; Here R0 contains SS$_ code in hi order..
      51    0C A2    D0  0C96  3141         MOVL    MSCP$L_BYTE_CNT(R2),R1                ; Get # bytes actually transferred.
                         0C9A  3142
                         0C9A  3143 TRANSFER_SHIFT:
                         0C9A  3144
 50   50    F0 8F    79  0C9A  3145         ASHQ    #-16,R0,R0                            ; Shift into proper position for IOSB.
                         0C9F  3146
                         0C9F  3147 NORMAL_TRANSFEREND:
                         0C9F  3148
   04 09 A2   03     E1  0C9F  3149         BBC     #MSCP$V_EF_EOT, -                     ; Is tape in the EOT region?
                         0CA4  3150                 MSCP$B_FLAGS(R2), 65$                 ; Branch if tape not in EOT.
                         0CA4  3151         ASSUME  MT$V_EOT  GE 16
      46 A3   04     88  0CA4  3152         BISB    #<MT$M_EOT @ -16>, -                  ; Else, set EOT DEVDEPEND position bit.
                         0CA8  3153                 UCB$L_DEVDEPEND+2(R3)
          0D 50      E9  0CA8  3154 65$:    BLBC    R0, 70$                               ; Branch if already returning an error.
 0A A2    0400 8F    B1  0CAB  3155         CMPW    #<MSCP$M_SC_EOT -                     ; Was a EOT subcode returned on a
                         0CB1  3156                  +MSCP$K_ST_SUCC>, -                  ; success command status?
                         0CB1  3157                 MSCP$W_STATUS(R2)
             05      12  0CB1  3158         BNEQ    70$                                   ; Branch if not EOT.
      50  0878 8F    B0  0CB3  3159         MOVW    #SS$_ENDOFTAPE, R0                    ; Else, return EOT status.
                         0CB8  3160
          00B0 C3    D5  0CB8  3161 70$:    TSTL    UCB$L_RECORD(R3)                      ; Previously at BOT?
             04      12  0CBC  3162         BNEQ    75$                                   ; Branch if not previously at BOT.
      40 A5   20     88  0CBE  3163         BISB    #CDRP$M_DENSCK, -                     ; Else, set density check required flag.
                         0CC2  3164                 CDRP$L_DUTUFLAGS(R5)
 00B0 C3    1C A2    D0  0CC2  3165 75$:    MOVL    MSCP$L_POSITION(R2), -                ; Update tape position information.
                         0CC8  3166                 UCB$L_RECORD(R3)
                         0CC8  3167
                         0CC8  3168 ; ----- BRB     FUNCTION_EXIT                         ; Go to common exit code.
                         0CC8  3169
                         0CC8  3170         .disable        lsb
```

```
                        0CC8          .SBTTL  FUNCTION_EXIT
                        0CC8
                        0CC8  ; FUNCTION_EXIT -
                        0CC8  ;
                        0CC8  ; INPUTS:
                        0CC8  ;       R0 => Final I/O status
                        0CC8  ;       R3 => UCB
                        0CC8  ;       R4 => PDT
                        0CC8  ;       R5 => CDRP
                        0CC8  ;
                        0CC8
                        0CC8
                        0CC8  FUNCTION_EXIT:
                        0CC8
                        0CC8          .IF     DF      TU_TRACE
                        0CC8          BSBW    TRACE_STATUS            ; Trace status.
                        0CC8          .ENDC
                        0CC8
    52    1C A5    D0   0CC8          MOVL    CDRP$L_MSG_BUF(R5),R2   ; R2 => end message.
             14    13   0CCC          BEQL    20$                     ; EQL implies no buffer.
             05    E0   0CCE          BBS     #MSCP$V_EF_ERLOG,-      ; Branch around if error log
       05 09 A2         0CD0                  MSCP$B_FLAGS(R2),10$    ;  message generated.
 0A 40 A5    02    E1   0CD3          BBC     #CDRP$V_ERLIP, -        ; If no ERLOG flag in End Message and
                        0CD8                  CDRP$L_DUTUFLAGS(R5), - ;  no remembered ERLIP, branch around.
                        0CD8                  20$
    40 A5    04    AA   0CD8  10$:    BICW    #CDRP$M_ERLIP, -        ; Clear error log in progress bit.
                        0CDC                  CDRP$L_DUTUFLAGS(R5)
   00000000'GF     16   0CDC          JSB     G^ERL$LOGSTATUS         ; Go log software status for errorlog.
                        0CE2
    D8 A5    50    D0   0CE2  20$:    MOVL    R0, CDRP$L_IOST1(R5)    ; Save final I/O status in CDRP.
                        0CE6          .IF     DF      TU_SEQCHK
                        0CE6          BSBB    SEQ_ENDCHECK            ; Check sequence on end.
                        0CE6          .ENDC
 32 40 A5    05    E5   0CE6          BBCC    #CDRP$V_DENSCK, -       ; Branch if density check not required
                        0CEB                  CDRP$L_DUTUFLAGS(R5), - ; and clear required flag.
                        0CEB                  30$
                        0CEB          ; Use a Set Unit Characteristics command to get the current density of
                        0CEB          ; the tape.  SUC is used instead of Get Unit Status because SUC is a
                        0CEB          ; sequential command.  This affords a better chance of coordinating
                        0CEB          ; with controller attempts to determine the density.  (Specificially,
                        0CEB          ; the HSC50 needs a sequential command here.)
                        0CEB          RESET_MSCP_MSG                  ; Else, setup to send another MSCP cmd.
    08 A2    0A    90   0CEE          MOVB    #MSCP$K_OP_STUNT, -     ; Make that command a set unit
                        0CF2                  MSCP$B_OPCODE(R2)       ; characteristics command.
 0E A2    00E0 C3  B0   0CF2          MOVW    UCB$W_UNIT_FLAGS(R3), - ; Must provide current unit flags
                        0CF8                  MSCP$W_UNT_FLGS(R2)     ; for SUC.
          00D8 C3  D0   0CF8          MOVL    UCB$L_MSCPDEVPARAM(R3),-; Must also provide device dependent
             1C A2      0CFC                  MSCP$L_DEV_PARM(R2)     ; parameters for SUC.
                        0CFE          SEND_MSCP_MSG                   ; Send the command.
                        0D01          IF_MSCP FAILURE, then=30$       ; Skip is get unit status failed.
 11 09 A2    02    E0   0D07          BBS     #MSCP$V_EF_PLS, -       ; Skip if correct tape position is
                        0D0C                  MSCP$B_FLAGS(R2), 30$   ; not known.
                        0D0C          ASSUME  MT$V_DENSITY GE 8       ; Otherwise, clear out previous
    44 A3    1F    8A   0D0C          BICB    #<MT$M_DENSITY a -8>, - ; density information.
                        0D10                  UCB$L_DEVDEPEND(R3)
    50    20 A2    3C   0D10          MOVZWL  MSCP$W_FORMAT(R2), R0   ; Get MSCP density value.
           F70E    30   0D14          BSBW    MSCPTOVMS_DENS          ; Convert density to VMS format.
   44 A3   05   08   50   F0  0D17  3229          INSV    R0, #MT$V_DENSITY, -     ; Store VMS density in UCB.
                              0D1D  3230                  #MT$S_DENSITY, -
                              0D1D  3231                  UCB$L_DEVDEPEND(R3)
                              0D1D  3232
                 F2E0'   30   0D1D  3233 30$:     BSBW    DUTU$DEALLOC_ALL         ; Free resources owned by this CDRP.
                              0D20  3234
           50   D8 A5    D0   0D20  3235          MOVL    CDRP$L_IOST1(R5), R0     ; Restore final I/O status.
           51   44 A3    D0   0D24  3236          MOVL    UCB$L_DEVDEPEND(R3),R1   ; Return to user I/O status block.
           52  00BC C3   D0   0D28  3237          MOVL    UCB$L_CDDB(R3),R2        ; R2 => CDDB.
                 00      E1   0D2D  3238          BBC     #CDDB$V_SNGLSTRM,-       ; See if in one at a time CDRP mode.
            0A 12 A2          0D2F  3239                  CDDB$W_STATUS(R2),100$   ;  If NOT branch around PUSHAB which
                              0D32  3240                                           ;  allows us to regain control after
                              0D32  3241                                           ;  ALT_REQCOM.
                 52      DD   0D32  3242          PUSHL   R2                       ; Save R2 => CDDB for after ALT_REQCOM.
                 54      DD   0D34  3243          PUSHL   R4                       ; Likewise save R4 => PDT.
          00000D42'EF    9F   0D36  3244          PUSHAB  110$                     ; Push address to which to return after
                              0D3C  3245                                           ;  ALT_REQCOM.
                              0D3C  3246 100$:
                              0D3C  3247          ALT_REQCOM
                              0D42  3248 110$:
                 54 8ED0      0D42  3249          POPL    R4                       ; Restore R4 => PDT.
                 53 8ED0      0D45  3250          POPL    R3                       ;  And R3 => CDDB.
               013B      31   0D48  3251          BRW     RESTART_NEXT_CDRP        ; Branch to code to restart next CDRP.
                              0D4B  3252
                              0D4B  3253          .IF     DF      TU_SEQCHK
                              0D4B  3254 ;+
                              0D4B  3255 ; SEQ_ENDCHECK - routine to check that commands end in sequence.
                              0D4B  3256 ;
                              0D4B  3257 ; Inputs:
                              0D4B  3258 ; R0 => Final I/O status
                              0D4B  3259 ; R3 => UCB
                              0D4B  3260 ; R5 => CDRP
                              0D4B  3261 ;
                              0D4B  3262 ; Outputs:
                              0D4B  3263 ;        All registers preserved.
                              0D4B  3264
                              0D4B  3265 SEQ_ENDCHECK:
                              0D4B  3266          PUSHL   R0                       ; Save R0 for later restore.
                              0D4B  3267          BBSC    #UCB$V_TU_OVRSQCHK,-     ; Branch around and clear bit if
                              0D4B  3268                  UCB$W_DEVSTS(R3),10$     ;  override specified.
                              0D4B  3269          EXTZV   #IRP$V_FCODE,-           ; Extract I/O function code.
                              0D4B  3270                  #IRP$S_FCODE,-
                              0D4B  3271                  CDRP$W_FUNC(R5),R0
                              0D4B  3272          BBC     R0,SEQ_MASK,10$          ; If non-Sequential I/O branch around.
                              0D4B  3273          CMPW    (SP),#SS$_ABORT          ; Is this an aborted command?
                              0D4B  3274          BEQL    50$                      ; Branch if aborted command.
                              0D4B  3275          EXTZV   #0,-                     ; Extract six bit index into array of
                              0D4B  3276                  #6,-                     ;  IRP sequence number slots.  R0 =
                              0D4B  3277                  UCB$B_TU_OLDINX(R3),R0   ;  index of oldest slot.
                              0D4B  3278          INCB    UCB$B_TU_OLDINX(R3)      ; Increment index.
                              0D4B  3279          CMPL    CDRP$L_SEQNUM(R5),-      ; Compare sequence number of this IRP to
                              0D4B  3280                  UCB$L_TU_SEQARY(R3)[R0]  ;  oldest outstanding sequence number.
                              0D4B  3281          BNEQ    99$                      ; Branch if terminating out of sequence.
                              0D4B  3282 10$:     POPL    R0                       ; Restore R0.
                              0D4B  3283          RSB                              ; Return to caller.
                              0D4B  3284
                              0D4B  3285          ; Process canceled, aborted command.
```

```
0D4B  3286 50$:    BSBW    REMOVE_SEQARY             ; Remove aborted command from list of
0D4B  3287                                           ;  commands.
0D4B  3288         BRB     10$                       ; Then exit this routine.
0D4B  3289
0D4B  3290 99$:    BUG_CHECK       TAPECLASS,FATAL ; Sequential command has been lost.
0D4B  3291         .ENDC
```

```
0D4B            .SBTTL  re-CONNECTION after VC error or failure
0D4B
0D4B    ; TU$CONNECT_ERR - Block of code invoked during the time that we
0D4B    ;       re-CONNECT to the intelligent controller following some disturbance
0D4B    ;       that caused dismanteling of the logical CONNECTION between the
0D4B    ;       class driver and the controller.  The ultimate purpose of the code
0D4B    ;       here is to locate all CDRP's relevant to this controller and place
0D4B    ;       them in the proper order into CDDB$L_RSTRTQFL.  Once
0D4B    ;       all the CDRP's are on this list we "execute" each of these CDRP's, one
0D4B    ;       by one, until they are all done.  When the last such CDRP is completed
0D4B    ;       we resume normal QIO processing.  This code works in cooperation with
0D4B    ;       code in FUNCTION_EXIT.
0D4B    ;
0D4B    ;       We are invoked here either by the Port Driver calling us at our error
0D4B    ;       entry point or by the Disk Class Driver branching here as a result of
0D4B    ;       deciding that the intelligent controller has gone ''insane''.
0D4B    ;
0D4B    ;       The actions herein taken are the following:
0D4B    ;
0D4B    ;       1.  We disable the Timeout Mechanism Routine wakeups by placing a
0D4B    ;               longword of all 1's in CRB$L_DUETIME.
0D4B    ;
0D4B    ;       2.  In order to prevent new CDRP's from starting up, we increment
0D4B    ;               UCB$W_RWAITCNT for each UCB associated with this controller.
0D4B    ;               This count is used to count the number of CDRP's associated
0D4B    ;               with a UCB that have run into resource wait situations.
0D4B    ;               Whenever this count is non-zero, new CDRP's are automatically
0D4B    ;               backed up onto the UCB$L_IRPQFL queue.  Incrementing this
0D4B    ;               count here, insures that it will not be run to zero and will
0D4B    ;               cause all new CDRP's to backup.
0D4B    ;
0D4B    ;       3.  We deallocate resources owned by the permanent CDRP used by the
0D4B    ;               Timeout Mechanism Routine.
0D4B    ;
0D4B    ;       4.  At the time that we are called here, our active CDRP's can be
0D4B    ;               found in one of the following places:
0D4B    ;
0D4B    ;               a) On the HIRT wait Q.  If here note that the associated UCB
0D4B    ;                  RWAITCNT has been bumped due to being on this list in
0D4B    ;                  addition to the bump given in step 2 above.
0D4B    ;
0D4B    ;               b) On the RDT resource wait Q.  Here also RWAITCNT has been
0D4B    ;                  bumped once to many times.
0D4B    ;
0D4B    ;               c) On the CDDB$L_CDRPQFL.  Here RWAITCNT is normal except for
0D4B    ;                  the bump given in step 1.
0D4B    ;
0D4B    ;               d) On some other resource wait Q (Flow control, message buffer,
0D4B    ;                  mapping resources, etc.). Here again RWAITCNT has been bumped
0D4B    ;                  once to much.
0D4B    ;
0D4B    ;               e) On the CDDB$L_RSTRTQ.  If here, the CONNECTION has failed
0D4B    ;                  while we were in the middle of cleaning up a previous
0D4B    ;                  CONNECTION failure.  The CDRP's here need no further
0D4B    ;                  gathering.
0D4B    ;
0D4B    ;               Our aim here is to gather all the active CDRP's onto the
```

```
0D4B  ;               CDDB$L_RSTRTQ.  To do this we search for them in the above
0D4B  ;               mentioned places in the order in which they were mentioned.
0D4B  ;               This order is important as will be explained below.
0D4B  ;
0D4B  ;       5.  Note here that at the time of the call to TU$CONNECT_ERR, we
0D4B  ;               may have been on the middle of MOUNT VERIFICATION.  In such
0D4B  ;               a case the particular volume would have been marked as
0D4B  ;               invalid and during re-CONNECTION we would not try to bring
0D4B  ;               the unit online.  Also we would have a set of inactive
0D4B  ;               (i.e. no resources allocated for them) CDRP's (IRP's) on
0D4B  ;               the MOUNT VERIFICATION QUEUE of the UCB and possibly one
0D4B  ;               MOUNT VERIFICATION specific CDRP active.  This all meshes
0D4B  ;               perfectly with our re-CONNECTION design.  The contents of
0D4B  ;               the MOUNT VERIFICATION QUEUE can be ignored.  The active
0D4B  ;               MOUNT VERIFICATION CDRP will be treated normally.  Its
0D4B  ;               I/O will be retried and will probably fail and MOUNT
0D4B  ;               VERIFICATION will re-submit it and it will wind up on the
0D4B  ;               normal UCB I/O QUEUE awaiting the RWAITCNT's going to zero.
0D4B  ;               After re-CONNECTION, it will start up normally and everything
0D4B  ;               should resume transparently.
0D4B  ;
0D4B  ;       6.  First we scan the HIRT wait Q and remove any CDRP's associated
0D4B  ;               with the current CDDB.  We do this first so that if perchance,
0D4B  ;               some of our CDRP's are here, they will not be selected
0D4B  ;               inadvertantly when the current HIRT owner is possibly killed.
0D4B  ;
0D4B  ;               This scan is done by going down the entire HIRT wait Q and
0D4B  ;               removing the 1st entry of ours that we find.  If in a pass
0D4B  ;               we DO remove an entry , then we go bask and scan from the
0D4B  ;               start of the Q.  When we make an entire pass without any hits,
0D4B  ;               we finish.  Note that when we remove an entry, we decrement
0D4B  ;               the RWAITCNT prior to calling INSERT_RSTRTQ to undo the bump
0D4B  ;               we gave in calling LOCK_HIRT.
0D4B  ;
0D4B  ;       7.  We scan the RDT resource wait Q.  Again we scan until we find our
0D4B  ;               first entry and after a removal we begin to scan from the
0D4B  ;               beginning.  Only a clean scan wnds the process.  Also we
0D4B  ;               must decrement RWAITCNT for each removal.
0D4B  ;
0D4B  ;       8.  We REMQUE each entry on CDDB$L_CDRPQFL and call INSERT_RSTRTQ
0D4B  ;               for each one.
0D4B  ;
0D4B  ;       9.  Here we should note that INSERT_RSTRTQ deallocates all resources
0D4B  ;               owned by a CDRP prior to inserting it in CDDB$L_RSTRTQ.
0D4B  ;               Because of this, the only CDRP's belonging to us that still
0D4B  ;               own RSPID's are the CDRP's which are on other resource wait
0D4B  ;               queues.  So here we scan the RDT looking for entries that
0D4B  ;               belong to us.  When we find one we REMQUE it, decrement its
0D4B  ;               RWAITCNT and call INSERT_RSTRTQ for it.  Note that this
0D4B  ;               deallocates its resources and as a result of this could cause
0D4B  ;               another of our CDRP's to receive these resources and proceed
0D4B  ;               up to the CDDB$L_CDRPQFL.  Therefore after a removal here,
0D4B  ;               we branch back to step 7 to safeguard against this possibility.
0D4B  ;               A complete scan of the RDT with no hits implies that we now
0D4B  ;               have gathered all our CDRP's and that we can continue.
0D4B  ;
0D4B  ;*********************************************************************
```

```
                    0D4B  3407 ;        9.  If the two counts above are equal, then we have all CDRP's on
                    0D4B  3408 ;                CDDB$L_RSTRTQFL.  No more CDRP's will trickle in so we clear
                    0D4B  3409 ;                CDDB$M_CDRPTRCKL in CDDB$W_STATUS.
                    0D4B  3410 ;
                    0D4B  3411 ;        10. We DISCONNECT the now dead connection and then re-CONNECT to
                    0D4B  3412 ;                establish a new channel to the MSCP server in the controller.
                    0D4B  3413 ;
                    0D4B  3414 ;        11. We are now ready to begin single stream execution of CDRPs, until
                    0D4B  3415 ;                exhaust the contents of the CDRP$L_RSTRTQFL.  However we
                    0D4B  3416 ;                want to guard against the possibility that a particular
                    0D4B  3417 ;                request (i.e. CDRP) may repeatedly hang a controller (i.e.
                    0D4B  3418 ;                cause a re-CONNECTION) and thereby prevent anything from
                    0D4B  3419 ;                getting through.  To deal with this we only retry a given
                    0D4B  3420 ;                request a fixed maximum number of times (MAX_RETRY).  The
                    0D4B  3421 ;                algorithm which resolves this retry logic dilemma relies
                    0D4B  3422 ;                on several data items in the CDDB:
                    0D4B  3423 ;
                    0D4B  3424 ;                        a) CDDB$L_RSTRTCDRP - the address of the CDRP that is
                    0D4B  3425 ;                            currently being processed in single stream mode if
                    0D4B  3426 ;                            we are in single stream mode.
                    0D4B  3427 ;
                    0D4B  3428 ;                        b) CDDB$B_RETRYCNT - the number of remaining retries
                    0D4B  3429 ;                            for the current CDRP being processes in single
                    0D4B  3430 ;                            stream mode if we are in single stream mode.
                    0D4B  3431 ;
                    0D4B  3432 ;                        c) CDDB$V_SNGLSTRM - bit in CDDB$W_STATUS which tells
                    0D4B  3433 ;                            us if we are in single stream mode.
                    0D4B  3434 ;
                    0D4B  3435 ;                The algorithm is as follows: If upon selecting the first CDRP
                    0D4B  3436 ;                on CDDB$L_RSTRTQFL, we find CDDB$V_SNGLSTRM clear, we merely
                    0D4B  3437 ;                set it and we can be assurred that this is the first time
                    0D4B  3438 ;                that we are attempting to retry this request in single stream
                    0D4B  3439 ;                mode.  This is so because the bit being clear implies either
                    0D4B  3440 ;                that this is the first re-CONNECTION since the system came up
                    0D4B  3441 ;                or that the last re-CONNECTION ran to completion thereby leaving
                    0D4B  3442 ;                the bit clear.  In this case we select this first CDRP, set
                    0D4B  3443 ;                CDDB$B_RETRYCNT to the maximum and establish this CDRP as the
                    0D4B  3444 ;                current one by storing its address in CDDB$L_RSTRTCDRP.
                    0D4B  3445 ;
                    0D4B  3446 ;                If however CDDB$V_SNGLSTRM is set upon selecting a CDRP, we
                    0D4B  3447 ;                must compare the CDRP address to the current value of
                    0D4B  3448 ;                CDDB$L_RSTRTCDRP.  If they are NOT equal, then again this is
                    0D4B  3449 ;                the first retry attempt for this CDRP and we merely set the
                    0D4B  3450 ;                CDDB$B_RETRYCNT to the maximum and store the CDRP in
                    0D4B  3451 ;                CDDB$L_RSTRTCDRP.  If the CDRP has the same address however,
                    0D4B  3452 ;                we must decrement one from the retry count and if it is not
                    0D4B  3453 ;                exhausted attempt to process the CDRP again.
                    0D4B  3454 ;
                    0D4B  3455 ;                Note this all works even though the address of a CDRP is not
                    0D4B  3456 ;                necessarily unique. That is, many I/O requests in the life of
                    0D4B  3457 ;                the system may occupy the same CDRP in virtual space.  However,
                    0D4B  3458 ;                once re-CONNECTION logic begins, it deals only with the CDRPs
                    0D4B  3459 ;                on the CDDB$L_RSTRTQFL.  This list never grows until re-
                    0D4B  3460 ;                CONNECTION is run to completion since all new IRPs are
                    0D4B  3461 ;                being backed up.  Therefore even though we may run repeated
                    0D4B  3462 ;                re-CONNECTIONs that do not run to completion but rather each
                    0D4B  3463 ;                causes the connection to go down, through all this the
```

```
                           0D4B ;                       CDDB$L_RSTRTQFL is always monotonically decreasing and no
                           0D4B ;                       new CDRPs are entered onto it that were not there at the time
                           0D4B ;                       that we began to process the first re-CONNECTION.  In a fixed
                           0D4B ;                       list of CDRPs which all exist at the same time, the address
                           0D4B ;                       is a unique descriptor.
                           0D4B ;
                           0D4B ;               12. Note that CDDB$M_SNGLSTRM in CDDB$W_STATUS acts as a flag to
                           0D4B ;                       FUNCTION_EXIT so that it can aid in the one at a time re-
                           0D4B ;                       execution of the CDRP's.
                           0D4B ;
                           0D4B ;               13. For debugging sake, we loop thru all UCB's and check that their
                           0D4B ;                       UCB$W_RWAITCNT values are all equal to 1.
                           0D4B ;                       Also for debugging sake we check that CDDB$L_CDRPQFL is
                           0D4B ;                       empty.
                           0D4B ;
                           0D4B ;               14. We REMQUE the 1st CDRP on CDDB$L_RSTRTQFL and branch to
                           0D4B ;                       TU_RESTARTIO to begin its execution.
                           0D4B ;
                           0D4B ; Inputs: (for TU$RE_SYNCH)
                           0D4B ;               R3 => CRB
                           0D4B
                           0D4B
                           0D4B TU$RE_SYNCH:
                           0D4B
      53    10 A3     D0   0D4B         MOVL    CRB$L_AUXSTRUC(R3),R3   ; R3 => CDDB.
      54    14 A3     D0   0D4F         MOVL    CDDB$L_PDT(R3),R4       ; R4 => PDT.
      26 A3    04     91   0D53         CMPB    #MSCP$K_CM_EMULA, -     ; If this is the MSCP server, the right
                           0D57                 CDDB$B_CNTRLMDL(R3)     ; resynch technique is DISCONNECT.
               0A     13   0D57         BEQL    RECONN_COMMON           ; So, skip the MRESET setup.
               10     A8   0D59         BISW    #CDDB$M_RESYNCH,-       ; Signal that we should reset
            12 A3          0D5B                 CDDB$W_STATUS(R3)       ;  intelligent controller.
               04     11   0D5D         BRB     RECONN_COMMON           ; Branch around to common code.
                           0D5F
                           0D5F ; Inputs: (for TU$CONNECT_ERR)
                           0D5F ;               R3 => CDT
                           0D5F ;               R4 => PDT
                           0D5F ;
                           0D5F
                           0D5F TU$CONNECT_ERR:
                           0D5F
      53    5C A3     D0   0D5F         MOVL    CDT$L_AUXSTRUC(R3),R3   ; R3 => CDDB.
                           0D63 RECONN_COMMON:
         3A A3        B6   0D63         INCW    CDDB$W_RSTRTCNT(R3)     ; Count number of times reconnected.
                      AA   0D66         BICW    #<CDDB$M_IMPEND -       ; Signal: no immediate command pending
                           0D67                  !CDDB$M_INITING -      ;               out of initialization
                           0D67                  !CDDB$M_SNGLSTRM -     ;               no single stream in progress
                           0D67                  !CDDB$M_RSTRTWAIT>,-   ;               not waiting to restart CDRPs
   12 A3    0107 8F        0D67                 CDDB$W_STATUS(R3)
                           0D6C
      50    18 A3     D0   0D6C         MOVL    CDDB$L_CRB(R3),R0       ; R0 => CRB.
      18 A0    01     CE   0D70         MNEGL   #1,CRB$L_DUETIME(R0)    ; Prevent Timeout Mechanism wakeups.
                           0D74
               08     A8   0D74         BISW    #CDDB$M_RECONNECT,-     ; Set bit meaning that we are in
            12 A3          0D76                 CDDB$W_STATUS(R3)       ;  the re-CONNECTING state.
                           0D78
   53   0000007C 8F   C3   0D78         SUBL3   #<UCB$L_CDDB_LINK -     ; Get "previous" UCB address in R1.
                           0D7F                  -CDDB$L_UCBCHAIN>, -
```

```
           51        0D7F                  R3, R1
                     0D80
 51   00C4 C1    D0  0D80  10$:   MOVL     UCB$L_CDDB_LINK(R1), R1 ; Chain to next UCB (if any).
             0A  13  0D85         BEQL     20$                     ; EQL implies no more UCB's here.
F4 68 A1     0A  E2  0D87         BBSS     #UCB$V_MSCP_WAITBMP, -  ; Only bump RWAITCNT once.  If already
                     0D8C                  UCB$W_DEVSTS(R1), 10$   ;  bumped, branch back.
        56 A1    B6  0D8C         INCW     UCB$W_RWAITCNT(R1)      ; Prevent new CDRP's from starting up.
             EF  11  0D8F         BRB      10$                     ; Go look for more UCB's.
                     0D91  20$:
                     0D91
                     0D91  ;
                     0D91  ; Now we are sure that no new CDRP's will start.
                     0D91  ;
                     0D91
          F26C'  30  0D91         BSBW     DUTU$DISCONNECT_CANCEL  ; Perform disconnect cancel cleanup.
                     0D94
                     0D94  ; Deallocate RSPID & message buffer on each of the CDDB perm. IRP/CDRP pairs.
                     0D94
 55   0194 C3    9E  0D94         MOVAB    CDDB$A_DAPCDRP(R3), R5  ; Get DAP permanent CDRP address.
          F264'  30  0D99         BSBW     DUTU$DEALLOC_RSPID_MSG  ; Deallocate its RSPID & msg. buf.
 55   00D0 C3    9E  0D9C         MOVAB    CDDB$A_PRMCDRP(R3), R5  ; Get permanent CDRP address.
          F25C'  30  0DA1         BSBW     DUTU$DEALLOC_RSPID_MSG  ; Deallocate its RSPID & msg. buf.
                     0DA4
                     0DA4  ;
                     0DA4  ;      Registers here are:
                     0DA4  ;               R3 => CDDB
                     0DA4  ;               R4 => PDT.
                     0DA4  ;
                     0DA4
                     0DA4  ; Locate and prepare for restarting all CDRPs currently waiting for a RSPID.
                     0DA4  ; Since the class driver allocates a RSPID as the first step in any function,
                     0DA4  ; CDRPs found now will not be holding any resources and will not be active.
                     0DA4  ; Since these CDRPs hold no resources, their cleanup will not cause any other
                     0DA4  ; waiting requests to become active.  (This fact is not currently used, but it
                     0DA4  ; might be useful.)
                     0DA4
 53   00F4 C3    D0  0DA4         MOVL     CDDB$L_CDT(R3), R3              ; Get CDT address.
                     0DA9
             51  D4  0DA9         CLRL     R1                              ; Set SCAN_RSPID_WAIT flag.
                     0DAB         SCAN_RSPID_WAIT -                        ; Use SCS service to scan RSPID
                     0DAB                  action = DUTU$RECONN_LOOKUP     ; wait queue.
                     0DB8                                                  ; DUTU$RECONN_LOOKUP is in
                     0DB8                                                  ; DUTUSUBS.
                     0DB8
                     0DB8  ; Remove all CDRPs on the active requests queue.  These CDRPs:
                     0DB8  ;   a. have outstanding requests in the intelligent controller,
                     0DB8  ;   b. suffered allocation failures due to a broken connection,
                     0DB8  ;   c. represent the request during which an "insane" controller was detected.
                     0DB8  ; In any case, these CDRPs are not on any resource wait queue and do not have
                     0DB8  ; their associated resource wait count bumped due to need for a resource.
                     0DB8
          F245'  30  0DB8         BSBW     DUTU$DRAIN_CDDB_CDRPQ           ; Cleanup active requests.
                     0DBB
                     0DBB  ; Now scan the entire Response-id Descriptor Table for any remaining CDRPs
                     0DBB  ; belonging to this connection.  Presumably these CDRPs are on a resource wait
                     0DBB  ; queue somewhere.  In addtion, releasing whatever resources such CDRPs hold
                     0DBB  ; may cause other waiting CDRPs to become active.  Therefore, after every CDRP
```

```
                        0DBB  ; is located and processed, the active CDRP queue must be scanned again.
                        0DBB
             51     D6  0DBB          INCL    R1                              ; Set SCAN_RDT flag.
                        0DBD          SCAN_RDT -                              ; Use SCS service to scan RDT.
                        0DBD                  action = DUTU$RECONN_LOOKUP     ; DUTU$RECONN_LOOKUP is in
                        0DCA                                                  ; DUTUSUBS.
                        0DCA
      53   5C A3    D0  0DCA          MOVL    CDT$L_AUXSTRUC(R3), R3          ; Restore the CDDB address.
                        0DCE
                        0DCE  RESTART_FIRST_CDRP:
                        0DCE
                        0DCE  ;
                        0DCE  ; We come here either by falling thru from above code or by branching here
                        0DCE  ;       from CALL_SEND_MSG_BUF when the last CDRP has trickled in.
                        0DCE  ;
                        0DCE  ;
                        0DCE  ; If here all CDRP's are in CDDB$L_RSTRTQFL.  So no more will trickle.
                        0DCE  ;       Clear bit that prevents CALL_SEND_MSG_BUF from doing its job.
                        0DCE  ;
                        0DCE  ; INPUTS:
                        0DCE  ;       R3 => CDDB
                        0DCE  ;       R4 => PDT
                        0DCE  ;
                        0DCE
                        0DCE
                        0DCE  ;
                        0DCE  ; Here we DISCONNECT the old connection.
                        0DCE  ;
                        0DCE
      55   00D0 C3  9E  0DCE          MOVAB   CDDB$A_PRMCDRP(R3),R5         ; Put R5 => CDRP for coming BSBWs.
           50   53  D0  0DD3          MOVL    R3,R0                         ; R0 => CDDB.
      53     24 A5  D0  0DD6          MOVL    CDRP$L_CDT(R5),R3             ; Set R3 => CDT.
12 A0      0080 8F  A8  0DDA          BISW    #CDDB$M_NOCONN, -             ; Set no connection active flag.
                        0DE0                  CDDB$W_STATUS(R0)
             04     E5  0DE0          BBCC    #CDDB$V_RESYNCH,-             ; Do NOT branch around if we were called
         1C  12 A0      0DE2                  CDDB$W_STATUS(R0),2$          ;  in order to re-synchronize.
      53     1C A3  D0  0DE5          MOVL    CDT$L_PB(R3),R3               ; R3 => Path Block for MRESET, etc.
                        0DE9          MRESET  PB$B_RSTATION(R3),#1          ; Force controller to reset itself.
                        0DF3          MSTART  PB$B_RSTATION(R3)             ; And force controller to restart itself.
             05         0E00          RSB                                   ; Kill this thread. Rely on Port
                        0E01                                                ;  Driver calling error routine as
                        0E01                                                ;  a result of MRESET to accomplish
                        0E01                                                ;  DISCONNECT and subsequent logic.
                        0E01  2$:
                        0E01          DISCONNECT      #DISCONNECT_REASON
                        0E0A
                        0E0A          PERMCDRP_TO_CDDB -                    ; Get CDDB address in R3.
                        0E0A                  cdrp=R5, cddb=R3
                        0E11
                        0E11  ;
                        0E11  ; Deallocate mapping resources
                        0E11  ; and queue mount verification requests for post processing
                        0E11  ; <<< The mount verification references have been commented out in the  >>>
                        0E11  ; <<< following lines.  This driver does not do mount verification.     >>>
                        0E11  ; <<< When it is taught to do mount verification, however, the comment- >>>
                        0E11  ; <<< ed lines MUST be restored.                                        >>>
                        0E11  ;
```

```
                          0E11
                          0E11              ; Any mapping resources still owned by CDRPs on the restart queue are
                          0E11              ; deallocated here.  This deallocation is delayed until after the
                          0E11              ; DISCONNECT (and possible MRESET) to prevent an "insane" controller
                          0E11              ; from continuing to transfer via possibly re-allocated mapping
                          0E11              ; resources.  The mount verification queueing is delayed because the
                          0E11              ; mount verification operation may be holding mapping resources.
                          0E11
            3C A3    9F   0E11              PUSHAB  CDDB$L_RSTRTQFL(R3)      ; Setup listhead address.
            3C A3    DD   0E14              PUSHL   CDDB$L_RSTRTQFL(R3)      ; Setup first CDRP address.
                          0E17
            55 8ED0       0E17  4$:         POPL    R5                       ; Get next CDRP address.
       6E    55      D1   0E1A              CMPL    R5, (SP)                 ; Is it the listhead?
             07      13   0E1D              BEQL    6$                       ; If yes, all deallocations are done.
          F1DE'      30   0E1F              BSBW    DUTU$DEALLOC_ALL         ; Free MAP resources owned by this CDRP.
             65      DD   0E22              PUSHL   (R5)                     ; Push next CDRP address.
                          0E24  ;<<<        BBC     #IRP$V_MVIRP, -          ; Is this a mount verification IRP?
                          0E24  ;<<<                CDRP$W_STS(R5), 4$       ; Branch if not an MV IRP.
                          0E24  ;<<<        REMQUE  (R5), R0                 ; Else, remove IRP/CDRP from restart
                          0E24  ;<<<        POST_CDRP status=SS$_MEDOFL      ; queue and send it to post processing.
             F1      11   0E24              BRB     4$                       ; Loop till all restart CDRPs are done.
                          0E26
             8E      D5   0E26  6$:         TSTL    (SP)+                    ; Clear listhead pointer from stack.
                          0E28
                          0E28              ; Deallocate mapping resources whose description is stored in the
                          0E28              ; CDDB permanent CDRP.  This information was placed there by
                          0E28              ; DUTU$INSERT_RESTARTQ when it discovered that the HIRT permanent CDRP
                          0E28              ; owned mapping resources.  In this way, another thread is allowed to
                          0E28              ; use the HIRT permanent CDRP while this connection is broken.
                          0E28
  55    00D0 C3      9E   0E28              MOVAB   CDDB$A_PRMCDRP(R3), R5   ; Get CDRP in R5.
          F1D0'      30   0E2D              BSBW    DUTU$DEALLOC_ALL         ; Free old HIRT MAP resources.
                          0E30                                               ;  the HIRT CDRP and whose ownership
                          0E30                                               ;  has been transferred here.
                          0E30
                          0E30
                          0E30  ;
                          0E30  ; re-CONNECT - Here we call an internal subroutine which:
                          0E30  ;
                          0E30  ;       1. Makes a connection to the MSCP server in the intelligent
                          0E30  ;               controller.
                          0E30  ;
                          0E30  ;       2. Sends an MSCP command to SET CONTROLLER CHARACTERISTICS.
                          0E30  ;
                          0E30  ;       3. Allocates an MSCP buffer and RSPID for our future use in
                          0E30  ;               connection management.
                          0E30  ;
                          0E30  ; Upon return R4 => PDT and R5 => CDRP.
                          0E30  ;
                          0E30
           F34E      30   0E30              BSBW    MAKE_CONNECTION          ; Call subroutine to connect.
                          0E33
                          0E33              PERMCDRP_TO_CDDB -               ; Get CDDB address in R3.
                          0E33                      cdrp=R5, cddb=R3
     50    18 A3     D0   0E3A              MOVL    CDDB$L_CRB(R3),R0        ; Get CRB address.
 1C A0   0EF0'CF     9E   0E3E              MOVAB   W^TU$TMR, -              ; Establish permanent timeout routine.
                          0E44                      CRB$L_TOUTROUT(R0)
```

```
              51   2A A3   3C  0E44  3692          MOVZWL  CDDB$W_CNTRLTMO(R3), R1 ; Get controller timeout interval.
    18 A0  00000000'GF   51   C1  0E48  3693          ADDL3   R1, G^EXE$GL_ABSTIM, -  ; Use that to set next timeout
                               0E51  3694                  CRB$L_DUETIME(R0)       ; wakeup time.
                               0E51  3695
                               0E51  3696          ; The normal MSCP timeout mechanism is now in effect.  Henceforth,
                               0E51  3697          ; no fork thread may use the CDDB permanent CDRP as a fork block.
                               0E51  3698
                               0E51  3699          ASSUME  CDDB$V_DAPBSY GE 8
           13 A3    04   88   0E51  3700          BISB    #<CDDB$M_DAPBSY a -8>, -; Set DAP CDRP in use flag.
                               0E55  3701                  CDDB$W_STATUS+1(R3)
           55   54 A3    D0   0E55  3702          MOVL    CDDB$L_DAPCDRP(R3), R5  ; Get DAP CDRP address.
                  F1A4'  30   0E59  3703          BSBW    DUTU$POLL_FOR_UNITS     ; Interrogate controller, poll for units.
                               0E5C  3704                                          ;   Returns R3 => CDDB, R5 => CDRP.
                               0E5C  3705
                               0E5C  3706          ; Now it is necessary to propogate all the connection dependent
                               0E5C  3707          ; information regarding the newly formed connection to the MSCP server
                               0E5C  3708          ; to all the UCB's in the primary chain for this CDDB.  At the same
                               0E5C  3709          ; time, every RWAITCNT value is tested to insure that it is consistant
                               0E5C  3710          ; with what would be expected based upon the various possible reasons
                               0E5C  3711          ; which cause it to be bumped.  This is merely a debugging exercise.
                               0E5C  3712          ; In END_SINGLE_STREAM, RWAITCNT will be reduced by one and the wait
                               0E5C  3713          ; count bumped flag will be cleared.
                               0E5C  3714
                               0E5C  3715          ; This loop also brings previously valid units online, an activity
                               0E5C  3716          ; which would be performed by mount verification if it existed.
                               0E5C  3717
                               0E5C  3718          ; This loop also initializes previously uninitialized trace tables.
                               0E5C  3719          ; This must be performed after the call to DUTU$POLL_FOR_UNITS.
                               0E5C  3720
           55   84 A3    9E   0E5C  3721          MOVAB   <CDDB$L_UCBCHAIN -      ; Setup "previous" UCB address.
                               0E60  3722                  -UCB$L_CDDB_LINK>(R3), -
                               0E60  3723                  R5
        55   00C4 C5    D0   0E60  3724 15$:     MOVL    UCB$L_CDDB_LINK(R5), R5 ; Link to next UCB.
                     10   13   0E65  3725          BEQL    30$                     ; Branch if no more UCBs to test.
                  F196'  30   0E67  3726          BSBW    DUTU$INIT_CONN_UCB      ; Setup connection dep. UCB fields.
                               0E6A  3727          .IF     DEFINED TO_TRACE
                               0E6A  3728          BSBW    TRACE_INIT              ; Init IRP trace table.
                               0E6A  3729          .ENDC
                  F193'  30   0E6A  3730          BSBW    DUTU$CHECK_RWAITCNT     ; Validate the wait count value.
       EE 64 A5   0B     E1   0E6D  3731          BBC     #UCB$V_VALID, -         ; If unit is not valid, all done
                               0E72  3732                  UCB$L_STS(R5), 15$      ; for now.
                   F4CB  30   0E72  3733          BSBW    BRING_UNIT_ONLINE       ; Else, bring the unit back online.
                     E9   11   0E75  3734          BRB     15$                     ; Loop through all UCBs.
                               0E77  3735
                               0E77  3736 30$:     ; If this driver performed mount verification, it would now be
                               0E77  3737          ; possible to execute requests on behalf of any pending mount
                               0E77  3738          ; verification threads.  Therefore, the CDDB$V_NOCONN bit is
                               0E77  3739          ; cleared here.
                               0E77  3740          ;
                               0E77  3741          ; Since all threads which use the DAP CDRP as a fork block are now
                               0E77  3742          ; completed, that block may now be used for DAP operations.
                               0E77  3743          ; Therefore, the DAP CDRP busy flags is cleared too.
                               0E77  3744
     12 A3   0480 8F     AA   0E77  3745          BICW    #<CDDB$M_NOCONN -       ; Clear no-connection and
                               0E7D  3746                    !CDDB$M_DAPBSY>, -    ; DAP-CDRP-busy flags.
                               0E7D  3747                  CDDB$W_STATUS(R3)
                               0E7D  3748
                   0E7D           ; Processing of the first CDRP in the restart queue is about to begin.
                   0E7D           ; The queue of active requests should be empty:  check it.  N.B. if
                   0E7D           ; volume revalidation were being performed by mount verification, the
                   0E7D           ; active request queue might not be empty and it would be necessary to
                   0E7D           ; synchronize with mount verification activities as is done in the
                   0E7D           ; disk class driver.
                   0E7D
                   0E7D           ASSUME  CDDB$L_CDRPQFL EQ 0
      53   63  D1  0E7D           CMPL    (R3), R3               ; Empty listheads point to themselves.
           04  13  0E80           BEQL    RESTART_NEXT_CDRP      ; EQL implies that all is correct.
                   0E82           BUG_CHECK       TAPECLASS,FATAL
                   0E86
                   0E86
                   0E86   RESTART_NEXT_CDRP:
                   0E86
                   0E86   ;
                   0E86   ; Here we attempt to initiate the first (i.e. next) CDRP on the restart queue.
                   0E86   ;       In order to prevent getting caught in an infinite loop trying to
                   0E86   ;       initiate an operation that the controller cannot complete for
                   0E86   ;       one reason or another, we maintain a retry count and the address
                   0E86   ;       of the CDRP that we are currently single streaming.
                   0E86   ;
                   0E86   ;       In the normal case this is an isolated re-CONNECTION and the
                   0E86   ;       first CDRP on the restart queue is a random CDRP.  We notice this
                   0E86   ;       by seeing that the address of our first CDRP is not equal to the
                   0E86   ;       current contents of CDDB$L_RSTRTCDRP.
                   0E86   ;
                   0E86   ;       In the other case the connection failed while we were in single
                   0E86   ;       stream mode and the CDRP which we happened to be processing is the
                   0E86   ;       same CDRP that now heads our restart queue.  In this case, before
                   0E86   ;       initiating the processing of this CDRP, we decrement 1 from the
                   0E86   ;       retry count and if it remains non-zero, we restart the CDRP
                   0E86   ;       processing.  If the decrementing results in a zero retry count,
                   0E86   ;       then we log the event and effectively abort the CDRP by branching to
                   0E86   ;       FUNCTION_EXIT with an appropriate error status.  FUNCTION_EXIT, due
                   0E86   ;       to the setting of the CDDB$M_SNGLSTRM bit will then start the
                   0E86   ;       processing of the next CDRP on the restart queue.
                   0E86   ;
                   0E86   ; We can arrive here either by falling through from the above code or via
                   0E86   ;       a branch from FUNCTION_EXIT.  In either case we have:
                   0E86   ;
                   0E86   ; INPUT:
                   0E86   ;       R3 => CDDB
                   0E86   ;
                   0E86
 55    3C B3   0F  0E86           REMQUE  @CDDB$L_RSTRTQFL(R3),R5 ; R5 => 1st CDRP on restart queue.
           2F  1D  0E8A           BVS     END_SINGLE_STREAM       ; VS implies restart was empty.
           00  E3  0E8C           BBCS    #CDDB$V_SNGLSTRM,-      ; Set bit and if clear, this is 1st
   1B  12  A3      0E8E                   CDDB$W_STATUS(R3),20$   ;  time here for this CDRP, so branch.
 34 A3     55  D1  0E91           CMPL    R5,CDDB$L_RSTRTCDRP(R3) ; See if same CDRP as last time.
           15  12  0E95           BNEQ    20$                     ; NEQ implies not the same.
      38 A3    97  0E97           DECB    CDDB$B_RETRYCNT(R3)     ; If same, decrement 1 from retries.
           18  12  0E9A           BNEQ    30$                     ; NEQ implies retries remaining.
                   0E9C
                   0E9C   ;
                   0E9C   ; ********************************Log this error.*******************************
                   0E9C   ;
```

```
                                  0E9C   3806
          50    00000054 8F   D0  0E9C   3807            MOVL    #SS$_CTRLERR,R0             ; Indicate appropriate error status.
                         51   D4  0EA3   3808            CLRL    R1                          ;  And set second part of I/O status.
               53    BC A5    D0  0EA5   3809            MOVL    CDRP$L_UCB(R5),R3           ; R3 => UCB.
                       FE1C   31  0EA9   3810            BRW     FUNCTION_EXIT
                                  0EAC   3811 20$:
               34 A3     55   D0  0EAC   3812            MOVL    R5,CDDB$L_RSTRTCDRP(R3) ; Establish new single stream CDRP.
                         02   90  0EB0   3813            MOVB    #MAX_RETRY,-                ; Establish fresh retry count.
                      38 A3       0EB2   3814                    CDDB$B_RETRYCNT(R3)
                                  0EB4   3815 30$:
               53    BC A5    D0  0EB4   3816            MOVL    CDRP$L_UCB(R5),R3           ; R3 => UCB.
                       F710   31  0EB8   3817            BRW     TU_RESTARTIO                ; Restart the CDRP.
                                  0EBB   3818
                                  0EBB   3819
                                  0EBB   3820 END_SINGLE_STREAM:
                                  0EBB   3821
                                  0EBB   3822 ;
                                  0EBB   3823 ; Here we want to resume normal operation and get each unit going.
                                  0EBB   3824 ;          To do this we pickup each UCB in turn and call SCS$UNSTALLUCB
                                  0EBB   3825 ;          for it.  This has the effect of starting up as many (perhaps all)
                                  0EBB   3826 ;          of the IRP's (that's right IRP's) as possible that may have
                                  0EBB   3827 ;          backed up on the UCB input queue while we were in single stream mode.
                                  0EBB   3828 ;          We then go on to the next UCB until we exhaust all UCB's connected
                                  0EBB   3829 ;          to this CDDB.
                                  0EBB   3830 ;
                                  0EBB   3831
               12 A3     01   AA  0EBB   3832            BICW    #CDDB$M_SNGLSTRM, -         ; Clear single streaming CDRPs flag.
                                  0EBF   3833                    CDDB$W_STATUS(R3)
               50    3A A3    3C  0EBF   3834            MOVZWL  CDDB$W_RSTRTCNT(R3), R0 ; Get current restart count.
               55    84 A3    9E  0EC3   3835            MOVAB   <CDDB$L_UCBCHAIN -          ; Setup "previous" UCB address.
                                  0EC7   3836                    -UCB$L_CDDB_LINK>(R3), -
                                  0EC7   3837                    R5
                                  0EC7   3838
             55    00C4 C5    D0  0EC7   3839 10$:       MOVL    UCB$L_CDDB_LINK(R5), R5 ; Point to next UCB.
                         1D   13  0ECC   3840            BEQL    30$                         ; Branch if no more UCBs to process.
          68 A5    0400 8F    AA  0ECE   3841            BICW    #UCB$M_MSCP_WAITBMP, -      ; Indicate RWAITCNT no longer bumped.
                                  0ED4   3842                    UCB$W_DEVSTS(R5)
                      56 A5   B7  0ED4   3843            DECW    UCB$W_RWAITCNT(R5)          ; Unbump wait count.
                      F126'   30  0ED7   3844            BSBW    DUTU$CHECK_RWAITCNT         ; Else, check wait count and
                         09   BB  0EDA   3845            PUSHR   #^M<R0,R3>                  ; Save restart cnt. and CDDB address.
                  00000000'GF 16  0EDC   3846            JSB     G^SCS$UNSTALLUCB            ; Start up IRPs on UCB.
                         09   BA  0EE2   3847            POPR    #^M<R0,R3>                  ; Restore restart cnt. and CDDB address.
               3A A3     50   B1  0EE4   3848            CMPW    R0, CDDB$W_RSTRTCNT(R3) ; Did the unstall cause a restart?
                         DD   13  0EE8   3849            BEQL    10$                         ; Branch if no restart was caused.
                              05  0EEA   3850            RSB                                 ; Else, discontinue this thread.
                                  0EEB   3851
               12 A3     08   AA  0EEB   3852 30$:       BICW    #CDDB$M_RECONNECT, -        ; Clear reconnect in progress bit.
                                  0EEF   3853                    CDDB$W_STATUS(R3)
                              05  0EEF   3854            RSB                                 ; Ta De, Ta De, that's all folks.
```

```
0EF0          .SBTTL  TU$TMR - Class Driver Timeout Mechanism Routine
0EF0
0EF0    ;+
0EF0    ; TU$TMR - Time out Mechanism Routine.  This routine is called
0EF0    ;  periodically whenever CRB$L_DUETIME becomes due.  At the time of a
0EF0    ;  periodic call to TU$TMR the Class Driver is in one of three states
0EF0    ;  with respect to the intelligent mass storage controller associated
0EF0    ;  with the CRB pointed at by R3.
0EF0    ;
0EF0    ;       1. State #1, the "normal" state for which this routine is optimized,
0EF0    ;          is characterized by the following two conditions:
0EF0    ;
0EF0    ;               a) One or more MSCP commands are outstanding to the controller.
0EF0    ;                  This is determined by having a NON-empty queue of CDRP's
0EF0    ;                  hanging off the CDDB.
0EF0    ;
0EF0    ;               b) The oldest outstanding command was initiated since the
0EF0    ;                  previous invocation of TU$TMR and is therefore not very
0EF0    ;                  old.  This is determined by comparing the RSPID of the
0EF0    ;                  currently oldest command to the RSPID of the oldest request
0EF0    ;                  at the time of the previous invocation.  If they are not
0EF0    ;                  equal then we are in State #1.
0EF0    ;
0EF0    ;       2. State #2 is characterized by having NO outstanding MSCP commands in
0EF0    ;          the controller.  This is determined by finding an empty CDRP queue
0EF0    ;          in the CDDB.
0EF0    ;
0EF0    ;       3. State #3 is the state where MSCP commands are outstanding and the
0EF0    ;          oldest one has been outstanding for at least one previous TU$TMR
0EF0    ;          invocation.
0EF0    ;
0EF0    ;  If we determine that we are in state #1, we simply record the RSPID of the
0EF0    ;   currently oldest outstanding MSCP command in CDDB$L_OLDRSPID and we initial-
0EF0    ;   ize CDDB$L_OLDCMDSTS to all 1's.  We then calculate a new due time,
0EF0    ;   place it in CRB$L_DUETIME and return to our caller, which results
0EF0    ;   in scheduling ourselves for the next invocation of TU$TMR.
0EF0    ;
0EF0    ;  States #2 and #3 share some common code.  In both cases we will issue an
0EF0    ;   IMMEDIATE command to the controller but for diverse reasons.  In the case
0EF0    ;   of state #2 it will be an effective NOP command that is only issued to
0EF0    ;   insure against the controller timing out the host (i.e. us) due to lack of
0EF0    ;   activity on our part.  In the case of state #3, the IMMEDIATE command will
0EF0    ;   be a "GET COMMAND STATUS" for the oldest outstanding MSCP command.
0EF0    ;
0EF0    ;  The common code they share consists of code to appropriate the pre-allocated
0EF0    ;   MSCP buffer pointed at by CDRP$L_MSG_BUF and to pick up the pre-allocated
0EF0    ;   RSPID identified by CDRP$L_RSPID.  BOth these items are located in
0EF0    ;   the permanent CDRP which is appended to the CDDB of this intelligent
0EF0    ;   controller.  Also at this time a new due time is calculated prior to
0EF0    ;   doing the DRIVER_SEND_MSG so that we will be able to time out the
0EF0    ;   Immediate command.  Then the code for these two states diverges for
0EF0    ;   a while to prepare distinct MSCP packets, do the SEND_MSG_BUF,
0EF0    ;   and in the case of state #3, to do some specific processing upon
0EF0    ;   receipt of the END PACKET for the IMMEDIATE command.  This processing
0EF0    ;   consists of insuring that the command status returned in the END PACKET
0EF0    ;   indicates progress being made on the oldest outstanding command; and also
0EF0    ;   of saving this received command status in the CDDB$L_OLDCMDSTS so as to
```

```
                      0EF0  ;      have it available at the next invocation, if this oldest command is still
                      0EF0  ;      outstanding.  Following this the two code paths converge to recycle the
                      0EF0  ;      received END PACKET for use as the next IMMEDIATE MSCP buffer and to also
                      0EF0  ;      recycle the RSPID by bumping its sequence number.
                      0EF0  ;
                      0EF0  ; INPUTS:
                      0EF0  ;         R3 => CRB of the intelligent disk controller
                      0EF0  ;
                      0EF0  ; OUTPUTS:
                      0EF0  ;         Registers R0 through R5 are all possibly modified.
                      0EF0  ;
                      0EF0
                      0EF0 TU$TMR:
                      0EF0          SETIPL  #IPL$_SCS                 ; After wakeup lower IPL.
   51    10 A3    D0  0EF3          MOVL    CRB$L_AUXSTRUC(R3),R1     ; R1 => CDDB.
                      0EF7
                      0EF7          ASSUME  CDDB$L_CDRPQFL   EQ   0
     51    61     D1  0EF7          CMPL    (R1),RT                   ; If =, then list of CDRP's is empty
           21     13  0EFA          BEQL    20$                       ; EQL means empty list of CDRP's,
                      0EFC                                            ;  which implies we are in State #2.
     50    61     D0  0EFC          MOVL    (R1),R0                   ; R0 => CDRP associated with "oldest"
                      0EFF                                            ;  outstanding MSCP command.
                      0EFF
        20 A0     D1  0EFF          CMPL    CDRP$L_RSPID(R0),-        ; Compare RSPID of oldest request to
        2C A1         0F02                  CDDB$L_OLDRSPID(R1)       ;  that of request current at time of
                      0F04                                            ;  previous invocation of TU$TMR.
           1C     13  0F04          BEQL    30$                       ; EQL implies State #3, i.e. current
                      0F06                                            ;  oldest has been around for awhile.
                      0F06
        20 A0     D0  0F06          MOVL    CDRP$L_RSPID(R0),-        ; State #1, we have a NEW oldest request
        2C A1         0F09                  CDDB$L_OLDRSPID(R1)       ;  so record its RSPID in CDDB field.
 30 A1     01     CE  0F0B          MNEGL   #1,CDDB$L_OLDCMDSTS(R1)   ; And initialize its associated status.
                      0F0F 10$:
   7E    2A A1    3C  0F0F          MOVZWL  CDDB$W_CNTRLTMO(R1),-(SP); Pickup controller delta.
           8E     C1  0F13          ADDL3   (SP)+,-                   ; Calculate delta time for next
 00000000'GF          0F15                  G^EXE$GL_ABSTIM,-         ;  periodic invocation of TU$TMR.
        18 A3         0F1A                  CRB$L_DUETIME(R3)
                  05  0F1C          RSB                               ; And return to caller.
                      0F1D
                      0F1D 20$:                                       ; If we are here, there are NO outstand-
                      0F1D                                            ;  ing requests in the controller since
                      0F1D                                            ;  CDRP list is empty.
           50     D4  0F1D          CLRL    R0                        ; R0 flagged to indicate State #2.
        2C A1     D4  0F1F          CLRL    CDDB$L_OLDRSPID(R1)       ; Set to impossible value to prevent
                      0F22                                            ;  inadvertent comparison error.
                      0F22
                      0F22 30$:                                       ; Common State #2, State #3 code path.
                      0F22                                            ; If here, for sure we will be issuing
                      0F22                                            ;  an immediate command to the controller.
                      0F22                                            ;  If we are in State #2, it will be a
                      0F22                                            ;  "GET UNIT STATUS" (NOP) command but
                      0F22                                            ;  if we are in State #3, it will be
                      0F22                                            ;  a "GET COMMAND STATUS" command.  For
                      0F22                                            ;  either case we begin the common setup.
                      0F22
                      0F22
   54    14 A1    D0  0F22          MOVL    CDDB$L_PDT(R1),R4         ; Setup for SEND_MSG_BUF, R4=>PDT.
 55   00D0 C1   9E  0F26             MOVAB   CDDB$A_PRMCDRP(R1),R5   ; R5 => CDRP appended to CDDB.
             01   E3  0F2B             BBCS    #CDDB$V_IMPEND,-        ; Branch if an immediate command is NOT
         03 12 A1       0F2D                     CDDB$W_STATUS(R1),40$   ;  pending. Also set bit to show that
                        0F30                                             ;  one WILL be pending momentarily.
           FE18   31  0F30             BRW     TU$RE_SYNCH             ; Bit set implies that an immediate
                        0F33                                             ;  "GET STATUS" type command has not
                        0F33                                             ;  completed in the timeout interval.
                        0F33                                             ;  So we goto resynchronization logic.
                        0F33
                        0F33     40$:
          7E   50   7D  0F33             MOVQ    R0, -(SP)               ; Save valuable registers.
                        0F36             INIT_MSCP_MSG                   ; Initalize buffer for MSCP message.
          50   8E   7D  0F39             MOVQ    (SP)+, R0               ; Restore valuable registers.
                        0F3C
               D1   10  0F3C             BSBB    10$                     ; Establish due time so as to be able
                        0F3E                                             ;  to timeout Immediate command.
               50   D5  0F3E             TSTL    R0                      ; Test for State #2 or State #3.
               09   12  0F40             BNEQ    50$                     ; NEQ implies State #3. Branch to handle it.
                        0F42
                        0F42     ; State #2 specific code.
                        0F42     ; Here we prepare the MSCP packet for the "GET UNIT STATUS" command for
                        0F42     ;       unit #0, which is an effective NOP command.  This is done to
                        0F42     ;       maintain minimum activity so that the controller will not time
                        0F42     ;       out the host (i.e. us). NOTE that since the MSCP buffer has been
                        0F42     ;       cleared above, there is no need to specify unit #0 in the command
                        0F42     ;       buffer.
                        0F42     ;
                        0F42
               03   90  0F42             MOVB    #MSCP$K_OP_GTUNT,-      ; Move in "GET UNIT STATUS" opcode.
             08 A2      0F44                     MSCP$B_OPCODE(R2)
                        0F46
                        0F46             SEND_MSCP_MSG DRIVER            ; Here we call to send the MSCP packet
                        0F49                                             ;  to the intelligent disk controller.
                        0F49
                        0F49                                             ; Return is experienced here after
                        0F49                                             ;  receipt of the END PACKET correspond-
                        0F49                                             ;  ing to the MSCP NOP sent above.  We
                        0F49                                             ;  regain control due to a callback
                        0F49                                             ;  from our own INPUT DISPATCHER
                        0F49                                             ;  ROUTINE.  Passed to us at this call-
                        0F49                                             ;  back are R2 => END PACKET, R3 => CRB,
                        0F49                                             ;  R4 => PDT and R5 => CDRP.
                        0F49                                             ; All we want to do is recycle the
                        0F49                                             ;  END PACKET for use as our next MSCP
                        0F49                                             ;  packet and recycle the RSPID.
                        0F49                                             ;  To do this we branch to common code.
               35   11  0F49             BRB     70$
                        0F4B
                        0F4B     50$:
                        0F4B
                        0F4B     ; State #3 specific code.
                        0F4B     ; Here we prepare the MSCP packet for a "GET COMMAND STATUS" command.
                        0F4B
     50     BC A0   D0  0F4B             MOVL    CDRP$L_UCB(R0),R0       ; R0 => UCB for oldest outstanding request.
                        0F4F
          00D4 C0   B0  0F4F             MOVW    UCB$W_MSCPUNIT(R0),-    ; Setup UNIT field.
             04 A2      0F53                     MSCP$W_UNIT(R2)
```

```
                      02   90  0F55  4027            MOVB    #MSCP$K_OP_GTCMD,-        ; Setup OPCODE field.
                   08 A2       0F57  4028                    MSCP$B_OPCODE(R2)
                               0F59  4029
                   2C A1   D0  0F59  4030            MOVL    CDDB$L_OLDRSPID(R1),-     ; Setup OUTSTANDING COMMAND REFERENCE
                   0C A2       0F5C  4031                    MSCP$L_OUT_REF(R2)        ;  field.
                               0F5E  4032
                               0F5E  4033            SEND_MSCP_MSG DRIVER              ; Here we call to send the MSCP packet
                               0F61  4034                                              ;  to the intelligent disk controller.
                               0F61  4035
                               0F61  4036                                              ; We experience return here upon receipt
                               0F61  4037                                              ;  of the END PACKET for the above ''GET
                               0F61  4038                                              ;  COMMAND STATUS'' command. We must make
                               0F61  4039                                              ;  sure that progress has indeed been
                               0F61  4040                                              ;  made on the outstanding command.  We
                               0F61  4041                                              ;  therefore compare the outstanding
                               0F61  4042                                              ;  command status returned in the END
                               0F61  4043                                              ;  PACKET to the previous value in CDDB
                               0F61  4044                                              ;  field CDDB$L_OLDCMDSTS.
                               0F61  4045                                              ; Here R2=>END PACKET, R3=>CRB, R4=>PDT
                               0F61  4046                                              ;  and R5=>CDRP.
                               0F61  4047
          51       10 A3   D0  0F61  4048            MOVL    CRB$L_AUXSTRUC(R3),R1     ; R1 => CDDB.
                   10 A2   D1  0F65  4049            CMPL    MSCP$L_CMD_STS(R2),-      ; Compare received outstanding command
                   30 A1       0F68  4050                    CDDB$L_OLDCMDSTS(R1)      ;  status to previous value.
                      0F   1F  0F6A  4051            BLSSU   60$                       ; LSSU implies progress made so branch.
                      0A   12  0F6C  4052            BNEQ    55$                       ; If not equal, progress went the
                               0F6E  4053                                              ;  wrong direction; a sure sign of
                               0F6E  4054                                              ;  an insane controller.
    10 A2     FFFFFFFF 8F  D1  0F6E  4055            CMPL    #-1, MSCP$L_CMD_STS(R2)   ; If equal to last time, is this the
                               0F76  4056                                              ;  multi-host busy somewhere else value?
                      03   13  0F76  4057            BEQL    60$                       ; Branch if it is busy somewhere else.
                    FDD0   31  0F78  4058 55$:       BRW     TU$RE_SYNCH               ; Anything else, implies no progress
                               0F7B  4059                                              ;  has been made.  So we goto
                               0F7B  4060                                              ;  re-synchronize with the intelligent
                               0F7B  4061                                              ;  disk controller and re-issue all
                               0F7B  4062                                              ;  outstanding commands.
                               0F7B  4063
                               0F7B  4064 60$:
                   10 A2   D0  0F7B  4065            MOVL    MSCP$L_CMD_STS(R2),-      ; Remember this received outstanding
                   30 A1       0F7E  4066                    CDDB$L_OLDCMDSTS(R1)      ;  command status for next time.
                               0F80  4067
                               0F80  4068 70$:                                         ; States #2 and #3 code paths merge here.
                               0F80  4069            RECYCH_MSG_BUF          ; Recycle END PACKET.
                               0F83  4070            RECYCL_RSPID                      ; Likewise the RSPID.
                               0F89  4071
          51       10 A3   D0  0F89  4072            MOVL    CRB$L_AUXSTRUC(R3),R1     ; R1 => CDDB.
                      02   AA  0F8D  4073            BICW    #CDDB$M_IMPEND,-          ; Indicate that immediate command is
                   12 A1       0F8F  4074                    CDDB$W_STATUS(R1)         ;  no longer pending.
                   F06C'   31  0F91  4075            BRW     DUTU$DODAP                ; Continue by doing DAP processing.
```

```
                    0F94          .SBTTL  TU$IDR - Class Driver Input Dispatch Routine
                    0F94
                    0F94  ;+
                    0F94  ; TU$IDR - Class Driver Input Dispatching Routine.  This routine is to
                    0F94  ;         the class driver what the Interrupt Service Routine is to a
                    0F94  ;         conventional driver.  We are called here by the Port Driver
                    0F94  ;         and we are passed the address of an END PACKET or an ATTENTION
                    0F94  ;         MESSAGE buffer.  By testing a bit in the ENDCODE field of the
                    0F94  ;         received buffer we determine which of the two has been received.
                    0F94  ;         For ATTENTION MESSAGES we immediately branch to ATTN_MSG.
                    0F94  ;
                    0F94  ;         For END PACKETs we first determine if the END PACKET is still of
                    0F94  ;         interest.  This is done by testing whether the COMMAND REFERENCE
                    0F94  ;         NUMBER returned in the END PACKET, interpreted as a RSPID, is
                    0F94  ;         still valid.  If not, we merely deallocate the END PACKET and
                    0F94  ;         return to our caller in the Port Driver.
                    0F94  ;
                    0F94  ;         If the END PACKET is still of interest then before dispatching
                    0F94  ;         to the code that originally issued the MSCP command for which we
                    0F94  ;         have just received the END PACKET, we first remove the
                    0F94  ;         CDRP associated with the command from the list of active CDRP's
                    0F94  ;         defined by the listhead located at CDDB$L_CDRPQFL.
                    0F94  ;
                    0F94  ; INPUTS:
                    0F94  ;         R1 =  Message Length
                    0F94  ;         R2 => END PACKET or ATTENTION MESSAGE BUFFER
                    0F94  ;         R3 => Connection Data Block
                    0F94  ;-
                    0F94
                    0F94  TU$IDR:
         07     E1  0F94          BBC     #MSCP$V_OP_END,-        ; Is this an ATTENTION MESSAGE
      08 A2         0F96                  MSCP$B_OPCODE(R2),-     ;  or an END PACKET;
         4A         0F98                  ATTN_MSG                ;  bit clear implies ATTENTION.
                    0F99
                    0F99
                    0F99  ;
                    0F99  ; Process command END MESSAGES
                    0F99  ;
                    0F99
         51     DD  0F99          PUSHL   R1                      ; Save message size.
   55    62     D0  0F9B          MOVL    MSCP$L_CMD_REF(R2), R5  ; Get RSPID from end message.
                    0F9E          FIND_RSPID_RDTE                 ; Lookup RDTE for RSPID.
         51   8ED0  0FA4          POPL    R1                      ; Restore message size.
      6B 50     E9  0FA7          BLBC    R0, FINISHED_WITH_MESSAGE ; Branch if error in RSPID.
   55    65     D0  0FAA          MOVL    RD$L_CDRP(R5),R5        ; R5 => CDRP.
50    24 A5     D0  0FAD          MOVL    CDRP$L_CDT(R5),R0       ; R0 => CDT.
50    5C A0     D0  0FB1          MOVL    CDT$L_AUXSTRUC(R0),R0   ; R0 => CDDB.
      2C A0     D1  0FB5          CMPL    CDDB$L_OLDRSPID(R0),-   ; See if oldest outstanding command has
         62         0FB8                  MSCP$L_CMD_REF(R2)      ;  this Command Reference Number.
         03     12  0FB9          BNEQ    20$                     ; If not, branch around.
      2C A0     D4  0FBB          CLRL    CDDB$L_OLDRSPID(R0)     ; Prevent inadvertent timeouts due to
                    0FBE                                          ;  reuse of RSPID in error situations.
                    0FBE  20$:    ASSUME  MSCP$K_LEN LT 32767
46 A5    51     B0  0FBE          MOVW    R1, CDRP$W_ENDMSGSIZ(R5); Save length of incomming packet.
1C A5    52     D0  0FC2          MOVL    R2, CDRP$L_MSG_BUF(R5)  ; Save address of incomming packet.
                    0FC6
   55    65     0F  0FC6          REMQUE  (R5),R5                 ; Remove R5=>CDRP from list.
```

```
                       0FC9              ASSUME  CDRP$V_CAND EQ 0
      0C 40 A5   E8    0FC9              BLBS    CDRP$L_DUTUFLAGS(R5), -  ; Has request been canceled?
                       0FCD                      30$                      ; If so, do cancel completion work.
0C CA A5    07   E0    0FCD      23$:    BBS     #IRP$V_DIAGBUF, -        ; Branch out of line if a diagnostic
                       0FD2                      CDRP$W_STS(R5), 50$      ; buffer was supplied.
                       0FD2
    53    10 A5  7D    0FD2      25$:    MOVQ    CDRP$L_FR3(R5), R3       ; Restore fork registers, R3 & R4.
          0C B5  17    0FD6              JMP     @CDRP$L_FPC(R5)          ; Dispatch to issuer of MSCP command
                       0FD9                                               ;  who will return to our caller.
                       0FD9
         F024'   30    0FD9      30$:    BSBW    DUTU$TEST_CANCEL_DONE    ; If this request completes a cancel
                       0FDC                                               ; operation, cleanup that operation.
            EF   11    0FDC              BRB     23$                      ; Branch back to normal flow.
                       0FDE
         F01F'   30    0FDE      50$:    BSBW    DUTU$DUMP_ENDMESSAGE     ; If diagnostic buffer, record MSCP
                       0FE1                                               ; end message sent in the buffer.
            EF   11    0FE1              BRB     25$                      ; Branch back to normal flow.
                       0FE3
                       0FE3
                       0FE3
                       0FE3      ;
                       0FE3      ; Process ATTENTION MESSAGES
                       0FE3      ;
                       0FE3
                       0FE3      ATTN_MSG:
            1E   BB    0FE3              PUSHR   #^M<R1,R2,R3,R4>         ; Save vital registers.
    53    5C A3  D0    0FE5              MOVL    CDT$L_AUXSTRUC(R3), R3   ; Get CDDB address.
         13'AF   9F    0FE9              PUSHAB  B^EXIT_ATTN_MSG          ; Make DISPATCH look like a BSBx.
                       0FEC              DISPATCH -                       ; Dispatch to attention message
                       0FEC                      MSCP$B_OPCODE(R2), -     ;  specific processing:
                       0FEC                      type=B, prefix=MSCP$K_OP , < -
                       0FEC                      <AVATN, UNIT_AVAILABLE_ATTN>, -
                       0FEC                      <DUPUN, DUPLICATE_UNIT_ATTN>, -
                       0FEC                      <ACPTH, ACCESS_PATH_ATTN>, -
                       0FEC                      >
                       0FF8      INV_ATTN_MSG:                            ; Process invalid ATTENTION MESSAGE.
            8E   D5    0FF8              TSTL    (SP)+                    ; Pop "return" address.
      50    0A   3C    0FFA              MOVZWL  #EMB$C_INVATT, R0        ; Invalid attention message type.
  00000000'GF    16    0FFD              JSB     G^ERL$LOG_TMSCP          ; Log incorrect TAPE MSCP message.
            1E   BA    1003              POPR    #^M<R1,R2,R3,R4>         ; Restore vital registers.
                       1005              DEALLOC_MSG_BUF_REG              ; Deallocate ATTN MSG buffer.
    53    5C A3  D0    1008              MOVL    CDT$L_AUXSTRUC(R3), R3   ; Get CDDB again.
    53    18 A3  D0    100C              MOVL    CDDB$L_CRB(R3), R3       ; From that get the CRB address.
          FD38   31    1010              BRW     TU$RE_SYNCH              ; Re-synchronize with controller.
                       1013
                       1013      EXIT_ATTN_MSG:
            1E   BA    1013              POPR    #^M<R1,R2,R3,R4>         ; Restore vital registers.
                       1015      FINISHED_WITH_MESSAGE:
                       1015              DEALLOC_MSG_BUF_REG              ; Deallocate ATTN MSG buffer.
                 05    1018              RSB                              ; Return to SCS caller.
```

```
                        1019  4185          .SBTTL Attention Message Processing
                        1019  4186          .SBTTL   - Process Unit Available Attention Message
                        1019  4187
                        1019  4188 ;++
                        1019  4189 ;
                        1019  4190 ; Functional Description:
                        1019  4191 ;
                        1019  4192 ;       This routine processes unit available attention messages.  If the
                        1019  4193 ;       available unit is already known in the I/O database, no action is
                        1019  4194 ;       taken.  If the available unit represents a second path to an already
                        1019  4195 ;       known unit, the I/O database is altered to show the alternate path
                        1019  4196 ;       availability.  If the available unit represents a totally new device,
                        1019  4197 ;       it is added to the I/O database.
                        1019  4198 ;
                        1019  4199 ; Inputs:
                        1019  4200 ;
                        1019  4201 ;       R1      attention message size
                        1019  4202 ;       R2      attention message address
                        1019  4203 ;       R3      CDDB address
                        1019  4204 ;
                        1019  4205 ; Outputs:
                        1019  4206 ;
                        1019  4207 ;       R0 - R5 destroyed
                        1019  4208 ;       All other registers preserved
                        1019  4209 ;--
                        1019  4210
                        1019  4211 UNIT_AVAILABLE_ATTN:
                        1019  4212
  03 12 A3    05   E0   1019  4213          BBS     #CDDB$V_POLLING, -              ; Is a poll for units in progress?
                        101E  4214                  CDDB$W_STATUS(R3), 90$          ; Branch if poll for units active.
          EFDF'    30   101E  4215          BSBW    DUTU$NEW_UNIT                   ; Process possible new unit.
                        1021  4216          .IF     DEFINED TU_TRACE
                        1021  4217          MOVL    R2, R5                          ; Copy UCB address.
                        1021  4218          BSBW    TRACE_INIT                      ; Initialize IRP trace table.
                        1021  4219          .ENDC
                   05   1021  4220 90$:     RSB
```

```
                         1022          .SBTTL   - Process Duplicate Unit Attention Message
                         1022
                         1022   ;++
                         1022   ;
                         1022   ; Functional Description:
                         1022   ;
                         1022   ;       This routine processes duplicate unit attention messages.
                         1022   ;       Notification of the condition is sent to the operator's console and
                         1022   ;       an entry is made in the error log.  If the unit described in the
                         1022   ;       message cannot be found, an invalid MSCP message error log entry is
                         1022   ;       generated.
                         1022   ;
                         1022   ; Inputs:
                         1022   ;
                         1022   ;       R1      attention message size
                         1022   ;       R2      attention message address
                         1022   ;       R3      CDDB address
                         1022   ;
                         1022   ; Outputs:
                         1022   ;
                         1022   ;       R0 - R5 destroyed
                         1022   ;       All other registers preserved
                         1022   ;--
                         1022
                         1022          .ENABLE LSB
                         1022
                         1022   DUPLICATE_UNIT_ATTN:
                         1022
          EFDB'    30    1022          BSBW    DUTU$LOOKUP_UCB         ; Locate UCB for this message.
      53   50      D0    1025          MOVL    R0, R3                  ; Setup UCB address.
           0C      13    1028          BEQL    90$                     ; If no UCB found, ignore the message.
          EFD3'    30    102A          BSBW    DUTU$SEND_DUPLICATE_UNIT; Send message to operator.
      50   06      3C    102D          MOVZWL  #EMB$C_DUPUN, R0        ; Setup duplicate unit error log code.
                         1030
                         1030   LOG_ATTENTION_MESSAGE:
 00000000'EF       16    1030          JSB     ERL$LOGMESSAGE          ; Error log attention message.
                   05    1036   90$:   RSB
                         1037
                         1037          .DISABLE LSB
```

```
                        1037            .SBTTL  - Process Access Path Attention Message
                        1037
                        1037    ;++
                        1037    ;
                        1037    ; Functional Description:
                        1037    ;
                        1037    ;       This routine processes access path attention messages.  If the access
                        1037    ;       path represents a second path to an already known unit, the I/O
                        1037    ;       database is altered to show the alternate path availability, and an
                        1037    ;       entry is made in the error log indicating receipt of the message.
                        1037    ;       If the unit described in the message cannot be found, an invalid MSCP
                        1037    ;       message error log entry is generated.
                        1037    ;
                        1037    ; Inputs:
                        1037    ;
                        1037    ;       R1      attention message size
                        1037    ;       R2      attention message address
                        1037    ;       R3      CDDB address
                        1037    ;
                        1037    ; Outputs:
                        1037    ;
                        1037    ;       R0 - R5 destroyed
                        1037    ;       All other registers preserved
                        1037    ;--
                        1037
                        1037    ACCESS_PATH_ATTN:
                        1037
        EFC6'   30      1037            BSBW    DUTU$SETUP_DUAL_PATH    ; Process possible dual path unit.
   53     50    D0      103A            MOVL    R0, R3                  ; Get UCB address.
          06    13      103D            BEQL    90$                     ; If no UCB found, ignore the message.
                05      103F            RSB                             ; Return w/o logging message, but
                        1040                                            ; leave message logging code in place
                        1040                                            ; just in case its needed.
   50     08    9A      1040            MOVZBL  #EMB$C_ACPTH, R0        ; Setup ERL$LOGMESSAGE code.
          EB    11      1043            BRB     LOG_ATTENTION_MESSAGE   ; Join common log message path.
                05      1045    90$:    RSB                             ; If no UCB, exit.
```

```
                               1046         .SBTTL  TU$DGDR - Data Gram Dispatch Routine
                               1046  ;
                               1046  ; Inputs:
                               1046  ;
                               1046  ;          R1 =  length of datagram
                               1046  ;          R2 => datagram
                               1046  ;          R3 => CDT
                               1046  ;          R4 => PDT
                               1046  ;
                               1046  TU$DGDR:
                               1046
        50    5C A3    D0      1046          MOVL    CDT$L_AUXSTRUC(R3),R0    ; R0 => CDDB
           55    53    D0      104A          MOVL    R3,R5                    ; Save pointer to CDT.
  50    0000007C 8F    C3      104D          SUBL3   #<UCB$L_CDDB_LINK -      ; Get "previous" UCB address in R3.
                               1054                   -CDDB$L_UCBCHAIN>, -
                 53            1054                  R0, R3
                               1055
      53    00C4 C3    D0      1055  10$:    MOVL    UCB$L_CDDB_LINK(R3), R3 ; Chain to next UCB (if any).
                 11    13      105A          BEQL    20$                      ; No more UCBs.
          00D4 C3      B1      105C          CMPW    UCB$W_MSCPUNIT(R3),-     ; See if datagram (error log packet)
            04 A2              1060                  MSCP$W_UNIT(R2)          ;   for this unit.
                 F1    12      1062          BNEQ    10$                      ; If not, branch abck to try next unit.
           50    02    3C      1064          MOVZWL  #EMB$C_TM,R0             ; Put type of message into R0.
       00000000'GF     16      1067          JSB     G^ERL$LOGMESSAGE         ; And call to log message.
                               106D  20$:
           53    55    D0      106D          MOVL    R5,R3                    ; Restore R3 => CDT.
      52    00B8 C4    C2      1070          SUBL    PDT$L_DGOVRHD(R4),R2     ; R2 => SCS header of datagram.
                               1075          QUEUE_DG_BUF                     ; Requeue datagram buffer.
                       05      1078          RSB                              ; Return to port.
```

```
                           1079         .SBTTL  INVALID_STS
                           1079
                           1079   ;+
                           1079   ; We come here if we get an invalid MSCP status.  We log the MSCP message
                           1079   ; and then RE-SYNCH the controller.
                           1079   ;
                           1079   ; Inputs:
                           1079   ;       R2 => MSCP packet
                           1079   ;       R3 => UCB
                           1079   ;       R4 => PDT
                           1079   ;       R5 => CDRP
                           1079   ;       CDRP$W_ENDMSGSIZ(R5) => length of MSCP packet with invalid status
                           1079   ;
                           1079   ;-
                           1079   INVALID_STS:
                           1079
         50    09     3C   1079         MOVZWL  #EMB$C_INVSTS,R0            ; Indicate type of record to log.
      51    46 A5     3C   107C         MOVZWL  CDRP$W_ENDMSGSIZ(R5), R1; Pickup length of faulty packet.
   53    00BC C3      D0   1080         MOVL    UCB$L_CDDB(R3),R3           ; R3 => CDDB for logging error.
    00000000'GF       16   1085         JSB     G^ERL$LOG_TMSCP             ; Log tape MSCP error.
          EF72'       30   108B         BSBW    DUTU$INSERT_RESTARTQ        ; Queue CDRP for retry.
      53    18 A3     D0   108E         MOVL    CDDB$L_CRB(R3),R3           ; R3 => CRB for re-SYNCH.
          FCB6        31   1092         BRW     TU$RE_SYNCH                 ; Zap controller.
```

```
1095            .SBTTL  TU_UNSOLNT
1095
1095    TU_UNSOLNT:
1095            BUG_CHECK       TAPECLASS,FATAL
1099
1099
1099            .IIF    DEFINED TU_TRACE, .PAGE
1099            .IF     DEFINED TU_TRACE
1099            .SBTTL  IRP Tracing Routines
1099            .SBTTL  - TRACE_INIT - Initialize trace table
1099    ;++
1099    ;
1099    ; TRACE_INIT - Initialize trace table
1099    ;
1099    ; Functional Description:
1099    ;
1099    ;       If the trace table is not initialized, initialize it.
1099    ;
1099    ; Inputs:
1099    ;
1099    ;       R5      UCB address.
1099    ;
1099    ; Implicit Inputs:
1099    ;
1099    ;       UCB$W_DEVSTS(R5)        UCB$V_TU_TRACEACT set if the trace table is
1099    ;                               initialized
1099    ;
1099    ; Outputs:
1099    ;
1099    ;       All registers preserved.
1099    ;
1099    ; Implicit Outputs:
1099    ;
1099    ;       UCB$W_DEVSTS(R5)        UCB$V_TU_TRACEACT is set if the trace table is
1099    ;                               successfully initialized
1099    ;       UCB$L_TRACEBEG(R5)      address of first IRP trace slot
1099    ;       UCB$L_TRACEPTR(R5)      address of first free IRP trace slot
1099    ;       UCB$L_TRACEND(R5)       address of first byte after IRP trace slots
1099    ;--
1099
1099    TRACE_SLOTS = 50                                ; Number of trace slots
1099    TRACE_SIZE = 96                                 ; Size of a trace slot
1099    TRACE_TBLSIZ = TRACE_SLOTS * TRACE_SIZE ; Size of the trace table
1099
1099            ASSUME  IRP$L_ARB+8 LE TRACE_SIZE
1099            ASSUME  <TRACE_SIZE & ^X1F> EQ 0
1099
1099    IRP$L_TU_TRCPTR = IRP$K_CD_LEN          ; Define a place to hold pointer to
1099    CDRP$L_TU_TRCPTR = CDRP$K_CD_LEN        ; trace slot
1099
1099            ASSUME  IRP$L_TU_TRCPTR+4 LE IRP$K_LENGTH
1099            ASSUME  CDRP$L_TU_TRCPTR-CDRP$L_IOQFL EQ IRP$L_TU_TRCPTR
1099
1099    TRACE_INIT:
1099
1099            BBS     #UCB$V_TU_TRACEACT, -   ; Branch if tracing is already
1099                    UCB$W_DEVSTS(R5), 90$   ; initialized.
```

```
1099                 PUSHR   #^M<R0,R1,R2,R3,R4,R5>   ; Save registers.
1099                 MOVZWL  #<TRACE_TBLSIZ+16>, R1   ; Get size of the trace table w/ header.
1099                 JSB     G^EXE$ALONONPAGED        ; Attempt to allocate pool.
1099                 BLBC    R0, 80$                  ; Branch if allocation failed.
1099                 CLRQ    (R2)+                    ; Initialize trace table header for SDA.
1099                 MOVW    R1, (R2)+                ; Save size.
1099                 MOVW    #DYN$C_CLASSDRV, (R2)+   ; Type.
1099                 CLRL    (R2)+                    ; Round header upto 16 byte boundary.
1099                 MOVL    R2, UCB$L_TRACEBEG(R5)   ; Save pointer to base of trace table.
1099                 MOVL    R2, UCB$L_TRACEPTR(R5)   ; Pointer to next area to use.
1099                 ADDL3   #TRACE_TBLSIZ, R2, -     ; Pointer to beyond end of trace table.
1099                         UCB$L_TRACEND(R5)
1099                 BISW    #UCB$M_TU_TRACEACT, -    ; Indicate Trace table inited.
1099                         UCB$W_DEVSTS(R5)
1099                 MOVC5   #0, (SP), #0, -          ; Zero trace table.
1099                         #TRACE_TBLSIZ, (R2)
1099
1099         80$:    POPR    #^M<R0,R1,R2,R3,R4,R5>   ; Restore registers.
1099         90$:    RSB                              ; Return
1099                 .PAGE
1099                 .SBTTL  - TRACE_IRP - Trace incomming IRP
1099         ;++
1099         ;
1099         ; TRACE_IRP - Trace incomming IRP
1099         ;
1099         ; Functional Description:
1099         ;
1099         ;       Called as a part of start I/O processing, this routine allocates a new
1099         ;       IRP trace slot and copies starting IRP contents into that slot.
1099         ;
1099         ;       IRP trace slots are 96 bytes long so that they line up nicely in
1099         ;       a dump.
1099         ;
1099         ; Inputs:
1099         ;
1099         ;       R3      IRP address
1099         ;       R5      UCB address
1099         ;
1099         ; Implicit Inputs:
1099         ;
1099         ;       UCB$W_DEVSTS(R5)        UCB$V_TU_TRACEACT set if IRP trace slots have
1099         ;                               been allocated
1099         ;       UCB$L_TRACEPTR(R5)      address of first free IRP trace slot
1099         ;       UCB$L_TRACEND(R5)       address of first byte after IRP trace slots
1099         ;       UCB$L_TRACEBEG(R5)      address of first IRP trace slot
1099         ;
1099         ; Outputs:
1099         ;
1099         ;       All registers preserved.
1099         ;
1099         ; Implicit Outputs:
1099         ;
1099         ;       UCB$L_TRACEPTR(R5)      updated
1099         ;       IRP$L_TU_TRCPTR(R3)     Address of IRP trace slot (for TRACE_STATUS)
1099         ;--
1099
1099         TRACE_IRP:
```

```
1099                    BBC     #UCB$V_TU_TRACEACT, -   ; If trace table not intialized,
1099                            UCB$W_DEVSTS(R5), 20$   ; exit immediately.
1099                    MOVQ    R0, -(SP)               ; Save R0 and R1.
1099                    MOVL    R3, R0                  ; Get IRP to trace in R0.
1099                    MOVL    UCB$L_TRACEPTR(R5), R1  ; Get address of next free trace slot.
1099                    CMPL    UCB$L_TRACEND(R5), R1   ; Check for end of trace table.
1099                    BGTR    10$                     ; Branch if not overflowed trace tbl.
1099                    MOVL    UCB$L_TRACEBEG(R5), R1  ; Else, reset to base of trace table.
1099     10$:           ADDL3   #TRACE_SIZE, R1, -      ; Setup next entry pointer.
1099                            UCB$L_TRACEPTR(R5)
1099
1099                    MOVL    R1, IRP$L_TU_TRCPTR(R3) ; Save trace slot addr at end of CDRP.
1099                    ASSUME  <TRACE_SIZE & 7> EQ 0
1099                    .REPEAT TRACE_SIZE / 8
1099                    MOVQ    (R0)+, (R1)+            ; Copy input IRP.
1099                    .ENDR
1099                    MOVL    IRP$L_TU_TRCPTR(R3), R1 ; Refresh R1 to trace slot beginning.
1099                    MOVL    R3, (R1)                ; Put IRP address in trace slot.
1099                    CLRL    4(R1)                   ; Clear field that will contain RSPID.
1099                    MNEGL   #1, IRP$L_ARB(R1)       ; Init field for I/O Status #1.
1099                    MNEGL   #1, IRP$L_ARB+4(R1)     ; Init field for I/O Status #2.
1099
1099                    MOVQ    (SP)+,R0                ; Restore R0 and R1.
1099     20$:           RSB
1099                    .PAGE
1099                    .SBTTL  - TRACE_STATUS - Trace final I/O request status
1099     ;++
1099     ;
1099     ; TRACE_STATUS - Trace final I/O request status
1099     ;
1099     ; Functional Description:
1099     ;
1099     ;       Copy final I/O status and RSPID into trace slot.
1099     ;
1099     ; Inputs:
1099     ;
1099     ;       R0      I/O status first longword
1099     ;       R3      UCB address
1099     ;       R5      CDRP address
1099     ;
1099     ; Implicit Inputs:
1099     ;
1099     ;       UCB$W_DEVSTS(R3)        UCB$V_TU_TRACEACT set if IRP trace slots have
1099     ;                               been allocated
1099     ;       CDRP$L_TU_TRCPTR(R5)    Address of IRP trace slot
1099     ;       UCB$L_DEVDEPEND(R3)     I/O status second longword
1099     ;
1099     ; Outputs:
1099     ;
1099     ;       All registers preserved.
1099     ;
1099     ; Implicit Outputs:
1099     ;
1099     ;       RSPID and final I/O status copies to IRP trace slot.
1099     ;--
1099
```

```
1099  4524 TRACE_STATUS:
1099  4525
1099  4526          BBC     #UCB$V_TU_TRACEACT, -   ; If trace table not initialized
1099  4527                  UCB$W_DEVSTS(R3), 30$   ; exit immediately.
1099  4528          PUSHL   R2                      ; Save register.
1099  4529          MOVL    CDRP$L_TU_TRCPTR(R5), R2; Get IRP trace slot address.
1099  4530          MOVL    CDRP$L_RSPID(R5), 4(R2) ; Save RSPID in trace.
1099  4531          MOVL    R0, IRP$L_ARB(R2)       ; Save I/O status.
1099  4532          MOVL    UCB$L_DEVDEPEND(R3), -  ;       ''
1099  4533                  IRP$L_ARB+4(R2)
1099  4534          POPL    R2                      ; Restore register.
1099  4535 30$:     RSB                             ; Return to caller.
1099  4536
1099  4537          .ENDC
1099  4538
1099  4539          .END
```

```
$$$                               = 00000020 R       04      CDDB$L_DDB                    = 0000001C
$$BASE                            = 00000040                 CDDB$L_OLDCMDSTS              = 00000030
$$BEGIN$$                         = 00000002                 CDDB$L_OLDRSPID               = 0000002C
$$DISPL                           = 00000043                 CDDB$L_PDT                    = 00000014
$$GENSW                           = 00000001                 CDDB$L_PRMUCB                   0000008C
$$HIGH                            = 00000042                 CDDB$L_RSTRTCDRP              = 00000034
$$LIMIT                           = 00000002                 CDDB$L_RSTRTQFL               = 0000003C
$$LOW                             = 00000040                 CDDB$L_SAVED_PC               = 00000044
$$MEDIA$$                         = 69A9504E                 CDDB$L_UCBCHAIN               = 00000048
$$MNSW                            = 00000001                 CDDB$M_DAPBSY                 = 00000400
$$MXSW                            = 00000001                 CDDB$M_IMPEND                 = 00000002
$$N$$                             = 0000004E                 CDDB$M_INITING                = 00000004
$$OP                              = 00000002                 CDDB$M_NOCONN                 = 00000080
$$$$$                             = 00000002                 CDDB$M_RECONNECT              = 00000008
$$TEMP$$                          = FFFFFFF7                 CDDB$M_RESYNCH                = 00000010
ACCESS_PATH_ATTN                    00001037 R       05      CDDB$M_RSTRTWAIT              = 00000100
ACP$ACCESS                          ********     X   05      CDDB$M_SNGLSTRM               = 00000001
ACP$DEACCESS                        ********     X   05      CDDB$Q_CNTRLID                = 00000020
ACP$MODIFY                          ********     X   05      CDDB$V_ALCLS_SET              = 00000006
ACP$MOUNT                           ********     X   05      CDDB$V_DAPBSY                 = 0000000A
ACP$READBLK                         ********     X   05      CDDB$V_IMPEND                 = 00000001
ACP$WRITEBLK                        ********     X   05      CDDB$V_INITING                = 00000002
ALLOC_DELTA                       = 00000001                 CDDB$V_POLLING                = 00000005
AT$_NULL                          = 00000005                 CDDB$V_RESYNCH                = 00000004
ATE_MSCPCODE                        00000002                 CDDB$V_SNGLSTRM               = 00000000
ATE_OFFSET                          00000000                 CDDB$W_CNTRLFLGS              = 00000028
ATE_SSCODE                          00000003                 CDDB$W_CNTRLTMO               = 0000002A
ATTN_MSG                            00000FE3 R       05      CDDB$W_RSTRTCNT               = 0000003A
AUTO_PACKACK                        0000048A R       05      CDDB$W_STATUS                 = 00000012
AVAILABLE_ABORT                     0000085F R       05      CDRP$B_CARCON                 = FFFFFFDC
AVAILABLE_CTRLERR                   0000085F R       05      CDRP$B_CD_TYPE                = 0000000A
AVAILABLE_DRVERR                    0000085F R       05      CDRP$B_EFN                    = FFFFFFC2
AVAILABLE_MEDOFL                    0000085F R       05      CDRP$B_FIPL                   = 0000000B
AVAILABLE_SEREX                     0000087E R       05      CDRP$B_IRP_TYPE               = FFFFFFAA
AVAILABLE_SUCC                      0000085F R       05      CDRP$B_PRI                    = FFFFFFC3
AVAIL_IVCMD                         00000857 R       05      CDRP$B_RMOD                   = FFFFFFAB
AVAIL_IVCMD_END                     0000085D R       05      CDRP$L_ABCNT                  = FFFFFFE0
BRING_UNIT_ONLINE                   00000340 R       05      CDRP$L_ARB                    = FFFFFFF8
BUG$_TAPECLASS                      ********     X   05      CDRP$L_AST                    = FFFFFFB0
CDDB$A_2PFKB                        00000174                 CDRP$L_ASTPRM                 = FFFFFFB4
CDDB$A_DAPCDRP                      00000194                 CDRP$L_BCNT                   = FFFFFFD2
CDDB$A_DAPIRP                       00000134                 CDRP$L_CDT                    = 00000024
CDDB$A_PRMCDRP                      000000D0                 CDRP$L_DIAGBUF                = FFFFFFEC
CDDB$A_PRMIRP                       00000070                 CDRP$L_DUTUFLAGS              = 00000040
CDDB$B_CNTRLMDL                   = 00000026                 CDRP$L_EXTEND                 = FFFFFFF4
CDDB$B_RETRYCNT                   = 00000038                 CDRP$L_FPC                    = 0000000C
CDDB$B_SYSTEMID                   = 0000000C                 CDRP$L_FR3                    = 00000010
CDDB$K_LENGTH                     = 00000070                 CDRP$L_IOQBL                  = FFFFFFA4
CDDB$L_ALLOCLS                    = 00000050                 CDRP$L_IOQFL                  = FFFFFFA0
CDDB$L_CANCLQBL                     000000B4                 CDRP$L_IOSB                   = FFFFFFC4
CDDB$L_CANCLQFL                     000000B0                 CDRP$L_IOST1                  = FFFFFFD8
CDDB$L_CDRPQFL                    = 00000000                 CDRP$L_IOST2                  = FFFFFFDC
CDDB$L_CDT                          000000F4                 CDRP$L_JNL_SEQNO              = FFFFFFE8
CDDB$L_CRB                        = 00000018                 CDRP$L_LBUFH_AD               = 0000002C
CDDB$L_DAPCDRP                    = 00000054                 CDRP$L_MEDIA                  = FFFFFFD8
CDDB$L_DAPCDT                       000001B8                 CDRP$L_MSG_BUF                = 0000001C
CDDB$L_DAPUCB                       00000150                 CDRP$L_OBCNT                  = FFFFFFE4
```

```
CDRP$L_PID                      = FFFFFFAC                      DPT$C_LENGTH                    = 00000038
CDRP$L_RSPID                    = 00000020                      DPT$C_VERSION                   = 00000004
CDRP$L_RWCPTR                   = 00000028                      DPT$INITAB                        00000038 R       04
CDRP$L_SEGVBN                   = FFFFFFE8                      DPT$M_NOUNLOAD                  = 00000004
CDRP$L_SEQNUM                   = FFFFFFF0                      DPT$M_SCS                       = 00000008
CDRP$L_SVAPTE                   = FFFFFFCC                      DPT$REINITAB                      00000078 R       04
CDRP$L_TT_TERM                  = FFFFFFDC                      DPT$TAB                           00000000 R       04
CDRP$L_UCB                      = FFFFFFBC                      DT$_TA78                        = 00000006
CDRP$L_WIND                     = FFFFFFB8                      DT$_TA81                        = 00000009
CDRP$M_DENSCK                   = 00000020                      DT$_TK50                        = 0000000A
CDRP$M_ERLIP                    = 00000004                      DT$_TU78                        = 00000005
CDRP$Q_NT_PRVMSK                = FFFFFFE0                      DT$_TU81                        = 00000008
CDRP$T_LBOFHNDL                 = 00000030                      DUPLICATE_UNIT_ATTN               00001022 R       05
CDRP$V_CAND                     = 00000000                      DUTU$CANCEL                       ********     X   05
CDRP$V_DENSCK                   = 00000005                      DUTU$CHECK_RWAITCNT               ********     X   05
CDRP$V_ERLIP                    = 00000002                      DUTU$CREATE_CDDB                  ********     X   05
CDRP$V_IVCMD                    = 00000008                      DUTU$DEALLOC_ALL                  ********     X   05
CDRP$W_ABCNT                    = FFFFFFE0                      DUTU$DEALLOC_RSPID_MSG            ********     X   05
CDRP$W_BCNT                     = FFFFFFD2                      DUTU$DISCONNECT_CANCEL            ********     X   05
CDRP$W_BOFF                     = FFFFFFD0                      DUTU$DODAP                        ********     X   05
CDRP$W_CDRPSIZE                 = 00000008                      DUTU$DRAIN_CDDB_CDRPQ             ********     X   05
CDRP$W_CHAN                     = FFFFFFC8                      DUTU$DUMP_ENDMESSAGE              ********     X   05
CDRP$W_ENDMSGSIZ                = 00000046                      DUTU$END                          ********     X   04
CDRP$W_FUNC                     = FFFFFFC0                      DUTU$GET_DEVTYPE                  ********     X   05
CDRP$W_IRP_SIZE                 = FFFFFFA8                      DUTU$INIT_CONN_UCB                ********     X   05
CDRP$W_OBCNT                    = FFFFFFE4                      DUTU$INIT_MSCP_MSG                ********     X   05
CDRP$W_STS                      = FFFFFFCA                      DUTU$INIT_MSCP_MSG_UNIT           ********     X   05
CDT$L_AUXSTRUC                  = 0000005C                      DUTU$INSERT_RESTARTQ              ********     X   05
CDT$L_PB                        = 0000001C                      DUTU$INTR_ACTION_N                ********     X   05
CLASS_DRVR_NAME                   0000015B R       05           DUTU$INTR_ACTION_XFER             ********     X   05
CLU$GL_ALLOCLS                    ********     X   05           DUTU$KILL_THIS_THREAD             ********     X   05
CONNECT_DELTA                   = 0000000A                      DUTU$LOG_IVCMD                    ********     X   05
CRB$L_AUXSTRUC                  = 00000010                      DUTU$LOOKUP_UCB                   ********     X   05
CRB$L_DUETIME                   = 00000018                      DUTU$L_CDDB_LISTHEAD              00000000
CRB$L_INTD                      = 00000024                      DUTU$NEW_UNIT                     ********     X   05
CRB$L_TOUTROUT                  = 0000001C                      DUTU$POLL_FOR_UNITS               ********     X   05
DC$_TAPE                        = 00000002                      DUTU$POST_CDRP                    ********     X   05
DDB$L_ACPD                      = 00000010                      DUTU$RECONN_LOOKUP                ********     X   05
DDB$L_ALLOCLS                   = 0000003C                      DUTU$RESET_MSCP_MSG               ********     X   05
DDB$L_CONLINK                   = 00000038                      DUTU$RESTORE_CREDIT               ********     X   05
DDB$L_DDT                       = 0000000C                      DUTU$SEND_DRIVER_MSG              ********     X   05
DDB$L_UCB                       = 00000004                      DUTU$SEND_DUPLICATE_UNIT          ********     X   05
DEV$M_AVL                       = 00040000                      DUTU$SEND_MSCP_MSG                ********     X   05
DEV$M_CLU                       = 00000001                      DUTU$SETUP_DUAL_PATH              ********     X   05
DEV$M_DIR                       = 00000008                      DUTU$TEST_CANCEL_DONE             ********     X   05
DEV$M_ELG                       = 00400000                      DUTU$UNITINIT                     ********     X   05
DEV$M_FOD                       = 00004000                      DYN$C_CDRP                      = 00000039
DEV$M_IDV                       = 04000000                      DYN$C_CRB                       = 00000005
DEV$M_MSCP                      = 00000020                      DYN$C_DDB                       = 00000006
DEV$M_NNM                       = 00000200                      DYN$C_DPT                       = 0000001E
DEV$M_ODV                       = 08000000                      DYN$C_ORB                       = 00000049
DEV$M_SDI                       = 00000010                      DYN$C_UCB                       = 00000010
DEV$M_SQD                       = 00000020                      EMB$C_ACPTH                     = 00000008
DEV$V_CDP                       = 00000003                      EMB$C_DUPUN                     = 00000006
DEV$V_FOR                       = 00000018                      EMB$C_INVATT                    = 0000000A
DEV$V_MNT                       = 00000013                      EMB$C_INVSTS                    = 00000009
DISCONNECT_REASON               = 00000001                      EMB$C_TM                        = 00000002
```

```
END_PACKACK                   00000792 R      05    IO$_SPACERECORD          = 00000009
END_SINGLE_STREAM             00000EBB R      05    IO$_UNLOAD               = 00000001
ERASEGAP_POST                 000008F4 R      05    IO$_VIRTUAL              = 0000003F
ERL$LOGMESSAGE                ********    X   05    IO$_WRITECHECK           = 0000000A
ERL$LOGSTATUS                 ********    X   05    IO$_WRITELBLK            = 00000020
ERL$LOG_TMSCP                 ********    X   05    IO$_WRITEMARK            = 0000001C
EXE$FORK                      ********    X   05    IO$_WRITEOF              = 00000028
EXE$GL_ABSTIM                 ********    X   05    IO$_WRITEPBLK            = 0000000B
EXE$GQ_SYSTIME                ********    X   05    IO$_WRITEVBLK            = 00000030
EXE$INSIOQ                    ********    X   05    IOC$ALTREQCOM              ********    X   05
EXE$ONEPARM                   ********    X   05    IOC$GL_TU_CDDB             ********    X   06
EXE$SETMODE                   ********    X   05    IOC$MNTVER                 ********    X   05
EXE$ZEROPARM                  ********    X   05    IOC$RETURN                 ********    X   05
EXIT_ATTN_MSG                 00001013 R      05    IPL$_SCS                 = 00000008
FINISHED_WITH_MESSAGE         00001015 R      05    IRP$B_CARCON             = 0000003C
FKB$K_LENGTH                = 00000018              IRP$B_EFN                = 00000022
FUNCTAB_LEN                 = 00000088              IRP$B_PRI                = 00000023
FUNCTION_EXIT                 00000CC8 R      05    IRP$B_RMOD               = 0000000B
HOST_TIMEOUT                = 0000001E              IRP$B_TYPE               = 0000000A
HSTIMEOUT_ARRAY               0000017B R      05    IRP$K_LENGTH             = 000000C4
INI$BRK                       ********    X   05    IRP$L_ABCNT              = 00000040
INITIAL_CREDIT              = 0000000A              IRP$L_ARB                = 00000058
INITIAL_DG_COUNT            = 00000002              IRP$L_AST                = 00000010
INIT_IMMED_DELTA            = 0000001E              IRP$L_ASTPRM             = 00000014
INIT_TIMEOUT                  00000158 R      05    IRP$L_BCNT               = 00000032
INVALID_STS                   00001079 R      05    IRP$L_CDT                = 00000084
INV_ATTN_MSG                  00000FF8 R      05    IRP$L_DIAGBUF            = 0000004C
IO$V_CLSEREXCP              = 00000009              IRP$L_EXTEND             = 00000054
IO$V_DATACHECK              = 0000000E              IRP$L_FQFL               = 00000060
IO$V_INHRETRY               = 0000000F              IRP$L_IOQBL              = 00000004
IO$V_NOWAIT                 = 00000007              IRP$L_IOQFL              = 00000000
IO$V_REVERSE                = 00000006              IRP$L_IOSB               = 00000024
IO$_ACCESS                  = 00000032              IRP$L_IOST1              = 00000038
IO$_ACPCONTROL              = 00000038              IRP$L_IOST2              = 0000003C
IO$_AVAILABLE               = 00000011              IRP$L_JNL_SEQNO          = 00000048
IO$_CREATE                  = 00000033              IRP$L_MEDIA              = 00000038
IO$_DEACCESS                = 00000034              IRP$L_OBCNT              = 00000044
IO$_DELETE                  = 00000035              IRP$L_PID                = 0000000C
IO$_DSE                     = 00000015              IRP$L_SEGVBN             = 00000048
IO$_ERASETAPE               = 00000006              IRP$L_SEQNUM             = 00000050
IO$_MODIFY                  = 00000036              IRP$L_SVAPTE             = 0000002C
IO$_MOUNT                   = 00000039              IRP$L_TT_TERM            = 0000003C
IO$_NOP                     = 00000000              IRP$L_UCB                = 0000001C
IO$_PACKACK                 = 00000008              IRP$L_WIND               = 00000018
IO$_READLBLK                = 00000021              IRP$Q_NT_PRVMSK          = 00000040
IO$_READPBLK                = 0000000C              IRP$S_FCODE              = 00000006
IO$_READVBLK                = 00000031              IRP$V_DIAGBUF            = 00000007
IO$_RECAL                   = 00000003              IRP$V_FCODE              = 00000000
IO$_REWIND                  = 00000024              IRP$V_PHYSIO             = 00000008
IO$_REWINDOFF               = 00000022              IRP$W_ABCNT              = 00000040
IO$_SENSECHAR               = 0000001B              IRP$W_BCNT               = 00000032
IO$_SENSEMODE               = 00000027              IRP$W_BOFF               = 00000030
IO$_SETCHAR                 = 0000001A              IRP$W_CHAN               = 00000028
IO$_SETMODE                 = 00000023              IRP$W_FUNC               = 00000020
IO$_SKIPFILE                = 00000025              IRP$W_OBCNT              = 00000044
IO$_SKIPRECORD              = 00000026              IRP$W_SIZE               = 00000008
IO$_SPACEFILE               = 00000002              IRP$W_STS                = 0000002A
```

```
LOCAL_DEVICE                        0000056D R     05      MSCP$L_CMD_STS                 = 00000010
LOG_ATTENTION_MESSAGE               00001030 R     05      MSCP$L_DEV_PARM                = 0000001C
MAKE_CONNECTION                     00000181 R     05      MSCP$L_MAXQTREC                = 00000024
MASKM                             = 00000008               MSCP$L_MEDIA_ID                = 0000001C
MASKL                             = 04000000               MSCP$L_OUT_REF                 = 0000000C
MAX_RETRY                         = 00000002               MSCP$L_POSITION                = 0000001C
MIN_SEND_CREDIT                   = 00000002               MSCP$L_RCSKIPED                = 0000000C
MSCP$B_BUFFER                     = 00000010               MSCP$L_REC_CNT                 = 0000000C
MSCP$B_CNT_ALCS                   = 00000004               MSCP$L_TMGP_CNT                = 00000010
MSCP$B_FLAGS                      = 00000009               MSCP$L_TMSKIPED                = 00000010
MSCP$B_OPCODE                     = 00000008               MSCP$M_MD_CLSEX                = 00002000
MSCP$K_CM_EMULA                   = 00000004               MSCP$M_MD_COMP                 = 00004000
MSCP$K_CM_HSC50                   = 00000001               MSCP$M_MD_DLEOT                = 00000080
MSCP$K_CM_RC25                    = 00000003               MSCP$M_MD_EXCLU                = 00000020
MSCP$K_CM_TU81                    = 00000005               MSCP$M_MD_IMMED                = 00000040
MSCP$K_CM_UDA50                   = 00000002               MSCP$M_MD_OBJCT                = 00000004
MSCP$K_CM_UDA52                   = 00000006               MSCP$M_MD_REVRS                = 00000008
MSCP$K_LEN                        = 00000030               MSCP$M_MD_REWND                = 00000002
MSCP$K_MXCMDLEN                   = 00000024               MSCP$M_MD_SEREC                = 00000100
MSCP$K_OP_ACPTH                   = 00000042               MSCP$M_MD_UNLOD                = 00000010
MSCP$K_OP_AVAIL                   = 00000008               MSCP$M_SC_EOT                  = 00000400
MSCP$K_OP_AVATN                   = 00000040               MSCP$M_ST_MASK                 = 0000001F
MSCP$K_OP_COMP                    = 00000020               MSCP$M_TF_800                  = 00000001
MSCP$K_OP_DUPUN                   = 00000041               MSCP$M_TF_GCR                  = 00000004
MSCP$K_OP_ERASE                   = 00000012               MSCP$M_TF_PE                   = 00000002
MSCP$K_OP_ERGAP                   = 00000016               MSCP$M_UF_VSMSU                = 00000020
MSCP$K_OP_GTCMD                   = 00000002               MSCP$M_UF_WRTPH                = 00002000
MSCP$K_OP_GTUNT                   = 00000003               MSCP$M_UF_WRTPS                = 00001000
MSCP$K_OP_ONLIN                   = 00000009               MSCP$G_CNT_ID                  = 00000014
MSCP$K_OP_READ                    = 00000021               MSCP$Q_TIME                    = 00000014
MSCP$K_OP_REPOS                   = 00000025               MSCP$Q_UNIT_ID                 = 00000014
MSCP$K_OP_STCON                   = 00000004               MSCP$S_ST_MASK                 = 00000005
MSCP$K_OP_STUNT                   = 0000000A               MSCP$V_CF_MLTHS                = 00000002
MSCP$K_OP_WRITE                   = 00000022               MSCP$V_EF_EOT                  = 00000003
MSCP$K_OP_WRITM                   = 00000024               MSCP$V_EF_ERLOG                = 00000005
MSCP$K_SC_DLATE                   = 00000001               MSCP$V_EF_PLS                  = 00000002
MSCP$K_SC_ODDBC                   = 00000002               MSCP$V_MD_CLSEX                = 0000000D
MSCP$K_ST_ABRTD                   = 00000002               MSCP$V_MD_COMP                 = 0000000E
MSCP$K_ST_AVLBL                   = 00000004               MSCP$V_MD_DLEOT                = 00000007
MSCP$K_ST_BOT                     = 0000000D               MSCP$V_MD_IMMED                = 00000006
MSCP$K_ST_CNTLR                   = 0000000A               MSCP$V_MD_SEREC                = 00000008
MSCP$K_ST_COMP                    = 00000007               MSCP$V_OP_END                  = 00000007
MSCP$K_ST_DATA                    = 00000008               MSCP$V_SC_ALONL                = 00000008
MSCP$K_ST_DRIVE                   = 0000000B               MSCP$V_SC_DUPUN                = 00000007
MSCP$K_ST_FMTER                   = 0000000C               MSCP$V_SC_INOPR                = 00000006
MSCP$K_ST_HSTBF                   = 00000009               MSCP$V_ST_MASK                 = 00000000
MSCP$K_ST_ICMD                    = 00000001               MSCP$V_TF_800                  = 00000000
MSCP$K_ST_LED                     = 00000013               MSCP$V_TF_GCR                  = 00000002
MSCP$K_ST_OFFLN                   = 00000003               MSCP$V_TF_PE                   = 00000001
MSCP$K_ST_PLOST                   = 00000011               MSCP$V_UF_VSMSU                = 00000005
MSCP$K_ST_PRESE                   = 00000012               MSCP$V_UF_WRTPH                = 0000000D
MSCP$K_ST_RDTRN                   = 00000010               MSCP$V_UF_WRTPS                = 0000000C
MSCP$K_ST_SUCC                    = 00000000               MSCP$W_CNT_FLGS                = 0000000E
MSCP$K_ST_TAPEM                   = 0000000E               MSCP$W_CNT_TMO                 = 00000010
MSCP$K_ST_WRTPR                   = 00000006               MSCP$W_FORMAT                  = 00000020
MSCP$L_BYTE_CNT                   = 0000000C               MSCP$W_FORMENU                 = 00000024
MSCP$L_CMD_REF                    = 00000000               MSCP$W_HST_TMO                 = 00000010
```

```
MSCP$W_MODIFIER                 = 0000000A                     PACKACK_OFFLINE                   0000075C R      05
MSCP$W_NOISEREC                 = 00000028                     PACKACK_SUCC                      00000719 R      05
MSCP$W_SPEED                    = 00000022                     PB$B_RSTATION                   = 0000000C
MSCP$W_STATUS                   = 0000000A                     PDT$L_ALLOCMSG                  = 00000014
MSCP$W_UNIT                     = 00000004                     PDT$L_DEALRGMSG                 = 00000024
MSCP$W_UNT_FLGS                 = 0000000E                     PDT$L_DGOVRHD                   = 000000B8
MSCPTOSPEED                       00000445 R      05           PDT$L_MAPIRP                    = 00000034
MSCPTOVMS_DENS                    00000425 R      05           PDT$L_MRESET                    = 00000070
MSCP_SRVR_NAME                    0000016B R      05           PDT$L_MSTART                    = 00000074
MSG_BUF_FAILURE                   00000595 R      05           PDT$L_QUEUEDG                   = 0000003C
MT$CHECK_ACCESS                   ********    X   05           PDT$L_RCHMSGBUF                 = 00000044
MT$K_GCR_6250                   = 00000005                     PHYIO_VOLINV                      000005DE R      05
MT$K_NORMAL11                   = 0000000C                     PR$_IPL                         = 00000012
MT$K_NRZI_800                   = 00000003                     PRP_STCON_MSG                     0000028B R      05
MT$K_PE_1600                    = 00000004                     RD$L_CDRP                       = 00000000
MT$K_SPEED_DEF                  = 00000000                     RECONN_COMMON                     00000D63 R      05
MT$M_BOT                        = 00010000                     RECORD_COMMON                     000007AA R      05
MT$M_DENSITY                    = 00001F00                     RECORD_GETUNIT_CHAR               000007A3 R      05
MT$M_ENSEREXCP                  = 00000004                     RECORD_ONLINE                     00000795 R      05
MT$M_EOF                        = 00020000                     RECORD_SETUNIT_CHAR               00000795 R      05
MT$M_EOT                        = 00040000                     RECORD_STCON                      000002BF R      05
MT$M_HWL                        = 00080000                     RESTART_FIRST_CDRP                00000DCE R      05
MT$M_LOST                       = 00100000                     RESTART_NEXT_CDRP                 00000E86 R      05
MT$M_SEREXCP                    = 00000001                     REWIND_ABORT                      00000984 R      05
MT$S_DENSITY                    = 00000005                     REWIND_AVAIL                      00000984 R      05
MT$S_SPEED                      = 00000008                     REWIND_CTRLERR                    00000984 R      05
MT$V_BOT                        = 00000010                     REWIND_DRVERR                     00000984 R      05
MT$V_DENSITY                    = 00000008                     REWIND_END                        00000984 R      05
MT$V_ENSEREXCP                  = 00000002                     REWIND_FMTER                      00000984 R      05
MT$V_EOF                        = 00000011                     REWIND_IVCMD                      0000096A R      05
MT$V_EOT                        = 00000012                     REWIND_IVCMD_END                  00000970 R      05
MT$V_FORMAT                     = 00000004                     REWIND_OFFLINE                    00000984 R      05
MT$V_HWL                        = 00000013                     REWIND_PRESE                      00000984 R      05
MT$V_LOST                       = 00000014                     REWIND_SUCC                       00000974 R      05
MT$V_SPEED                      = 00000018                     SCS$ALLOC_RSPID                   ********    X   05
MT$V_SUP_GCR                    = 00000017                     SCS$CONNECT                       ********    X   05
MT$V_SUP_NRZI                   = 00000015                     SCS$DISCONNECT                    ********    X   05
MT$V_SUP_PE                     = 00000016                     SCS$FIND_RDTE                     ********    X   05
NOP_AVAIL                         000006B3 R      05           SCS$LKP_RDTCDRP                   ********    X   05
NOP_CTRLERR                       000006B3 R      05           SCS$LKP_RDTWAIT                   ********    X   05
NOP_DRVERR                        000006B3 R      05           SCS$RECYL_RSPID                   ********    X   05
NOP_IVCMD                         000006AB R      05           SCS$UNSTALLUCB                    ********    X   05
NOP_IVCMD_END                     000006B1 R      05           SENSEMODE_ONLINE                  00000B7E R      05
NOP_OFFLINE                       000006B3 R      05           SENSEMODE_RETURN                  00000B84 R      05
NOP_SUCC                          000006B3 R      05           SETMODE_ABORT                     00000A8E R      05
NORMAL_TRANSFEREND                00000C9F R      05           SETMODE_BEGIN_IVCMD               00000AB9 R      05
ORB$B_FLAGS                     = 0000000B                     SETMODE_CANCEL                    00000A9A R      05
ORB$B_TYPE                      = 0000000A                     SETMODE_CTRLERR                   00000A8E R      05
ORB$C_LENGTH                    = 00000058                     SETMODE_DRVERR                    00000A8E R      05
ORB$L_OWNER                     = 00000000                     SETMODE_IVCMD                     00000B40 R      05
ORB$M_PROT_16                   = 00000001                     SETMODE_IVCMD_END                 00000B46 R      05
ORB$W_PROT                      = 00000018                     SETMODE_OFFLINE                   00000A8E R      05
ORB$W_SIZE                      = 00000008                     SETMODE_ONLINE                    00000A9D R      05
PACKACK_CANCEL                    0000077F R      05           SETMODE_RETURN                    00000B4D R      05
PACKACK_GTUNT_SUCC                0000074B R      05           SETMODE_SUCC                      00000B4A R      05
PACKACK_IVCMD                     00000752 R      05           SET_CLEAR_SEX                     0000046A R      05
PACKACK_IVCMD_END                 00000758 R      05           SGN$GL_VMSD3                      ********    X   05
```

```
SKIP_ABORT                      00000A17 R      05    START_WRITEOF                   00000897 R      05
SKIP_AVAIL                      00000A17 R      05    START_WRITEPBLK                 00000B96 R      05
SKIP_BOT                        00000A29 R      05    TERMINATE_PENDING               000002FD R      05
SKIP_COMMON                     00000991 R      05    TRANSFER_BOT                    00000C48 R      05
SKIP_CTRLERR                    00000A2D R      05    TRANSFER_COMPERR                00000C96 R      05
SKIP_DRVERR                     00000A2D R      05    TRANSFER_CTRLERR                00000C5B R      05
SKIP_END                        00000A51 R      05    TRANSFER_DATA_ERROR             00000C96 R      05
SKIP_EOF                        00000A23 R      05    TRANSFER_EOF                    00000C42 R      05
SKIP_FMTER                      00000A2D R      05    TRANSFER_HOST_BUFFER_ERROR      00000C88 R      05
SKIP_IVCMD                      00000A0F R      05    TRANSFER_INVALID_COMMAND        00000C70 R      05
SKIP_IVCMD_END                  00000A15 R      05    TRANSFER_IVCMD_END              00000C76 R      05
SKIP_LEOT                       00000A2D R      05    TRANSFER_MEDOFL                 00000C7A R      05
SKIP_OFFLINE                    00000A17 R      05    TRANSFER_PLOST                  00000C3C R      05
SKIP_PLOST                      00000A1D R      05    TRANSFER_PRESE                  00000C51 R      05
SKIP_PRESE                      00000A17 R      05    TRANSFER_RTN_BCNT               00000C96 R      05
SKIP_SUCC                       00000A2D R      05    TRANSFER_RTN_RECLEN             00000C96 R      05
SPEEDTOMSCP                     00000430 R      05    TRANSFER_SHIFT                  00000C9A R      05
SS$_ABORT                     = 0000002C              TU$CONNECT_ERR                  00000D5F R      05
SS$_BUGCHECK                  = 000002A4              TU$DDT                          00000000 RG     05
SS$_CTRLERR                   = 00000054              TU$DGDR                         00001046 R      05
SS$_DATACHECK                 = 0000005C              TU$IDR                          00000F94 R      05
SS$_DATALATE                  = 00002274              TU$RE_SYNCH                     00000D4B R      05
SS$_DATAOVERUN                = 00000838              TU$TMR                          00000EF0 R      05
SS$_DEVOFFLINE                = 00000084              TU_ABSDENS                      00000400 R      05
SS$_DRVERR                    = 0000008C              TU_ABSPEED                      00000408 R      05
SS$_DUPUNIT                   = 000021C4              TU_BEGIN_IVCMD                  00000601 R      05
SS$_ENDOFFILE                 = 00000870              TU_CONTROLLER_INIT              000000C0 R      05
SS$_ENDOFTAPE                 = 00000878              TU_FUNCTABLE                    00000038 R      05
SS$_ENDOFVOLUME               = 000009A0              TU_MSCPDENS                     000003FD R      05
SS$_ILLIOFUNC                 = 000000F4              TU_REAL_STARTIO                 000005C5 R      05
SS$_IVBUFLEN                  = 0000034C              TU_REDO_IO                      00000601 R      05
SS$_MEDOFL                    = 000001A4              TU_RESTARTIO                    000005CB R      05
SS$_NORMAL                    = 00000001              TU_STARTIO                      00000598 R      05
SS$_PARITY                    = 000001F4              TU_UNSOLNT                      00001095 R      05
SS$_SERIOUSEXCP               = 000021D4              TU_VMSDENS                      000003F9 R      05
SS$_VOLINV                    = 00000254              UCB$B_DEVCLASS                = 00000040
SS$_WRITLCK                   = 0000025C              UCB$B_DEVTYPE                 = 00000041
START_AVAILABLE                 00000818 R      05    UCB$B_DIPL                    = 0000005E
START_DSE                       00000887 R      05    UCB$B_FIPL                    = 0000000B
START_ERASETAPE                 00000881 R      05    UCB$B_TYPE                    = 0000000A
START_NOP                       00000676 R      05    UCB$K_MSCP_TAPE_LENGTH        = 000000EC
START_PACKACK                   000006B8 R      05    UCB$K_TU_LENGTH               = 000000F8
START_READPBLK                  00000B9C R      05    UCB$L_2P_ALTUCB               = 000000A8
START_RECAL                     0000091C R      05    UCB$L_CDDB                    = 000000BC
START_REWIND                    0000091C R      05    UCB$L_CDDB_LINK               = 000000C4
START_REWINDOFF                 00000814 R      05    UCB$L_CDT                     = 000000C8
START_SENSECHAR                 00000B66 R      05    UCB$L_DEVCHAR                 = 00000038
START_SENSEMODE                 00000B66 R      05    UCB$L_DEVCHAR2                = 0000003C
START_SETCHAR                   00000A54 R      05    UCB$L_DEVDEPEND               = 00000044
START_SETMODE                   00000A59 R      05    UCB$L_IOQBL                   = 00000050
START_SKIPFILE                  00000987 R      05    UCB$L_IOQFL                   = 0000004C
START_SKIPRECORD                0000098D R      05    UCB$L_LINK                    = 00000030
START_SPACEFILE                 00000987 R      05    UCB$L_MEDIA_ID                = 0000008C
START_SPACERECORD               0000098D R      05    UCB$L_MSCPDEVPARAM            = 000000D8
START_UNLOAD                    00000814 R      05    UCB$L_PDT                     = 00000084
START_WRITECHECK                00000B87 R      05    UCB$L_RECORD                  = 000000B0
START_WRITEMARK                 00000897 R      05    UCB$L_STS                     = 00000064
```

```
UCB$L_TU_MAXWRCNT                 000000EC
UCB$M_BSY                       = 00000100
UCB$M_MSCP_INITING              = 00000200
UCB$M_MSCP_WAITBMP              = 00000400
UCB$M_MSCP_WRTP                 = 00002000
UCB$M_ONLINE                    = 00000010
UCB$M_TU_SEQNOP                 = 00000004
UCB$M_VALID                     = 00000800
UCB$Q_UNIT_ID                   = 000000CC
UCB$V_BSY                       = 00000008
UCB$V_MSCP_WAITBMP              = 0000000A
UCB$V_MSCP_WRTP                 = 0000000D
UCB$V_TU_SEQNOP                 = 00000002
UCB$V_VALID                     = 0000000B
UCB$W_DEVBUFSIZ                 = 00000042
UCB$W_DEVSTS                    = 00000068
UCB$W_MSCPUNIT                  = 000000D4
UCB$W_RWAITCNT                  = 00000056
UCB$W_SIZE                      = 00000008
UCB$W_STS                       = 00000064
UCB$W_TU_FORMAT                   000000F0
UCB$W_TU_NOISE                    000000F4
UCB$W_TU_SPEED                    000000F2
UCB$W_UNIT_FLAGS                = 000000E0
UNIT_AVAILABLE_ATTN               00001019 R      05
VALID_PACKACK                     0000078E R      05
VEC$L_INITIAL                   = 0000000C
VMSTOMSCP_DENS                    0000040C R      05
VOL_INVALID                       00000578 R      05
WRITM_ABORT                       000008F8 R      05
WRITM_AVAIL                       000008F8 R      05
WRITM_CTRLERR                     000008F8 R      05
WRITM_DATA_ERROR                  000008F8 R      05
WRITM_DRVERR                      000008F8 R      05
WRITM_END                         00000908 R      05
WRITM_FMTER                       000008F8 R      05
WRITM_IVCMD                       000008EA R      05
WRITM_IVCMD_END                   000008F0 R      05
WRITM_OFFLINE                     000008F8 R      05
WRITM_PRESE                       00000919 R      05
WRITM_SUCC                        000008F8 R      05
WRITM_WRITLCK                     000008F8 R      05
WTM_ERASE_COM                     0000089B R      05
XFER_IVCMD_END                    00000C3A R      05
```

```
                                                   +----------------+
                                                   ! Psect synopsis !
                                                   +----------------+

PSECT name                       Allocation           PSECT No.  Attributes
----------                       ----------           ---------  ----------
.  ABS  .                        00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                            000001F8  (  504.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
$$$200_TEMPLATE_UCB_01           000000F8  (  248.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC LONG
$$$200_TEMPLATE_ORB_01           00000058  (   88.)  03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC LONG
$$$105_PROLOGUE                  00000083  (  131.)  04 (  4.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
$$$115_DRIVER                    00001099  ( 4249.)  05 (  5.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC LONG
$$$220_DUTU_DATA_01              00000004  (    4.)  06 (  6.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC LONG
$$$220_DEVTYPE_TABLE_01          00000019  (   25.)  07 (  7.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC BYTE

                                                +------------------------+
                                                ! Performance indicators !
                                                +------------------------+

Phase                       Page faults    CPU Time        Elapsed Time
-----                       -----------    --------        ------------
Initialization                       30    00:00:00.04     00:00:01.28
Command processing                  109    00:00:00.47     00:00:02.87
Pass 1                             1050    00:00:43.71     00:02:52.53
Symbol table sort                     0    00:00:03.78     00:00:11.25
Pass 2                              411    00:00:10.19     00:00:37.49
Symbol table output                   1    00:00:00.40     00:00:02.65
Psect synopsis output                 0    00:00:00.03     00:00:00.03
Cross-reference output                0    00:00:00.00     00:00:00.00
Assembler run totals               1603    00:00:58.62     00:03:48.10

The working set limit was 3000 pages.
322530 bytes (630 pages) of virtual memory were used to buffer the intermediate code.
There were 190 pages of symbol table space allocated to hold 3488 non-local and 113 local symbols.
4539 source lines were read in Pass 1, producing 42 object records in Pass 2.
97 pages of virtual memory were used to define 89 macros.

                                               +--------------------------+
                                               ! Macro library statistics !
                                               +--------------------------+

Macro library name                             Macros defined
------------------                             --------------
_$255$DUA28:[DRIVER.OBJ]DUTULIB.MLB;1                16
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                       50
_$255$DUA28:[SYSLIB]STARLET.MLB;2                    12
TOTALS (all libraries)                               78

3948 GETS were required to define 78 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:TUDRIVER/OBJ=OBJ$:TUDRIVER MSRC$:TUDRIVER/UPDATE=(ENH$:TUDRIVER)+EXECML$/LIB+LIB$:DUTULIB/LIB
```

0117 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

TUDRIVER
LIS

XADRIVER
LIS

TSDRIVER
LIS